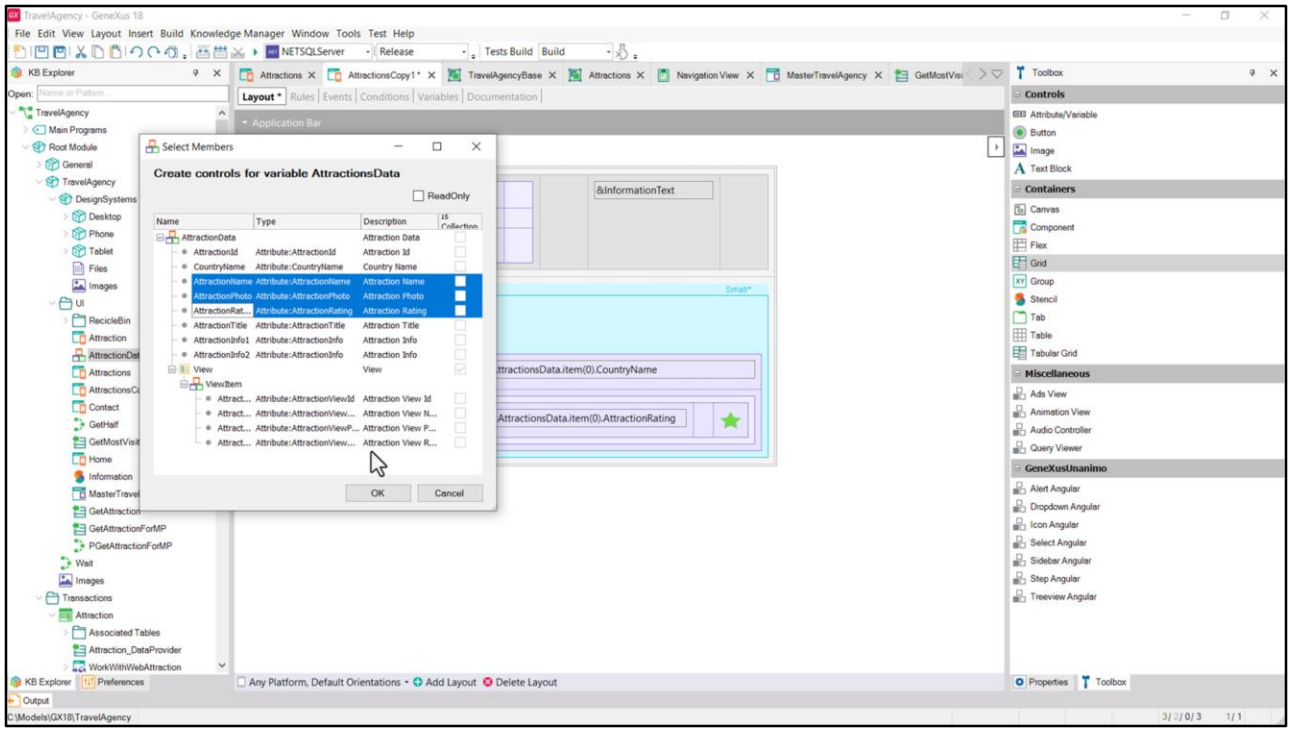
GeneXus by Globant

Attractions Panel: Carousel (Grid control) part 2

Cecilia Fernández

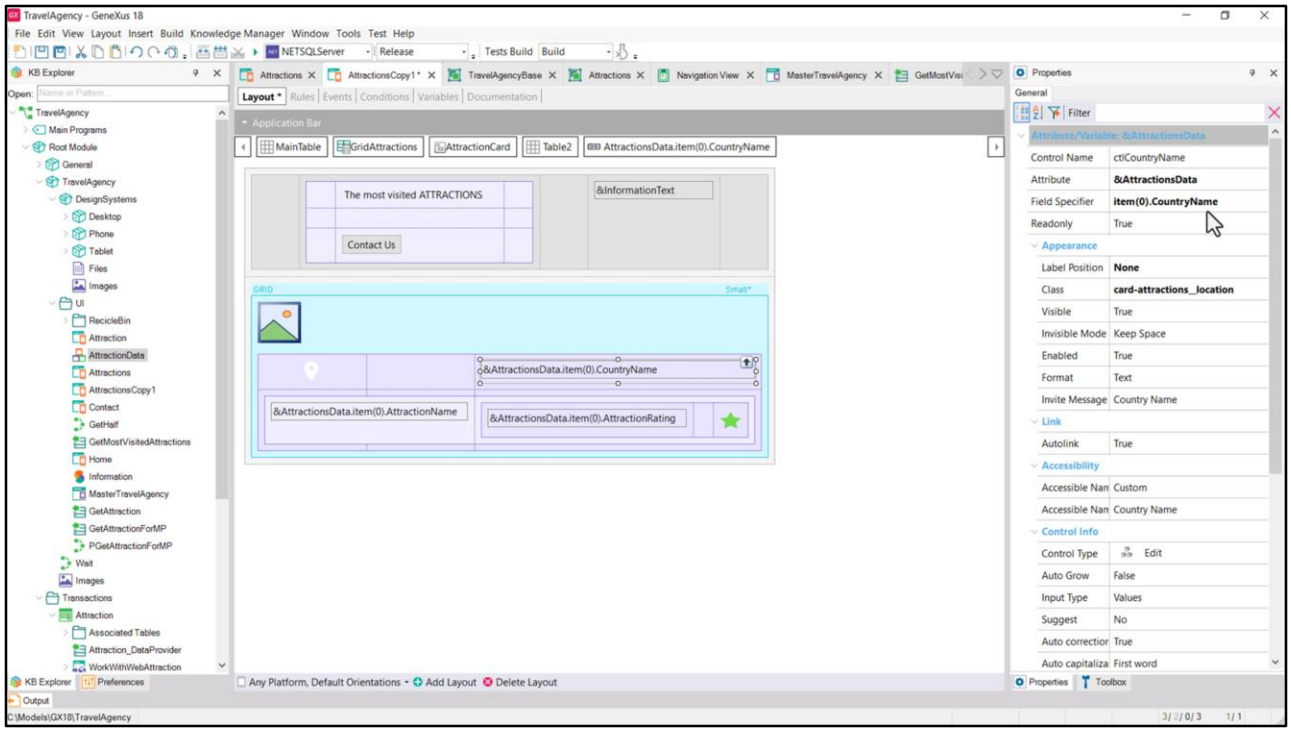


Bem, vamos continuar agora de onde paramos no vídeo anterior.

Como não mostrei a vocês, podem estar se perguntando como inseri todos esses campos de cada item da coleção.

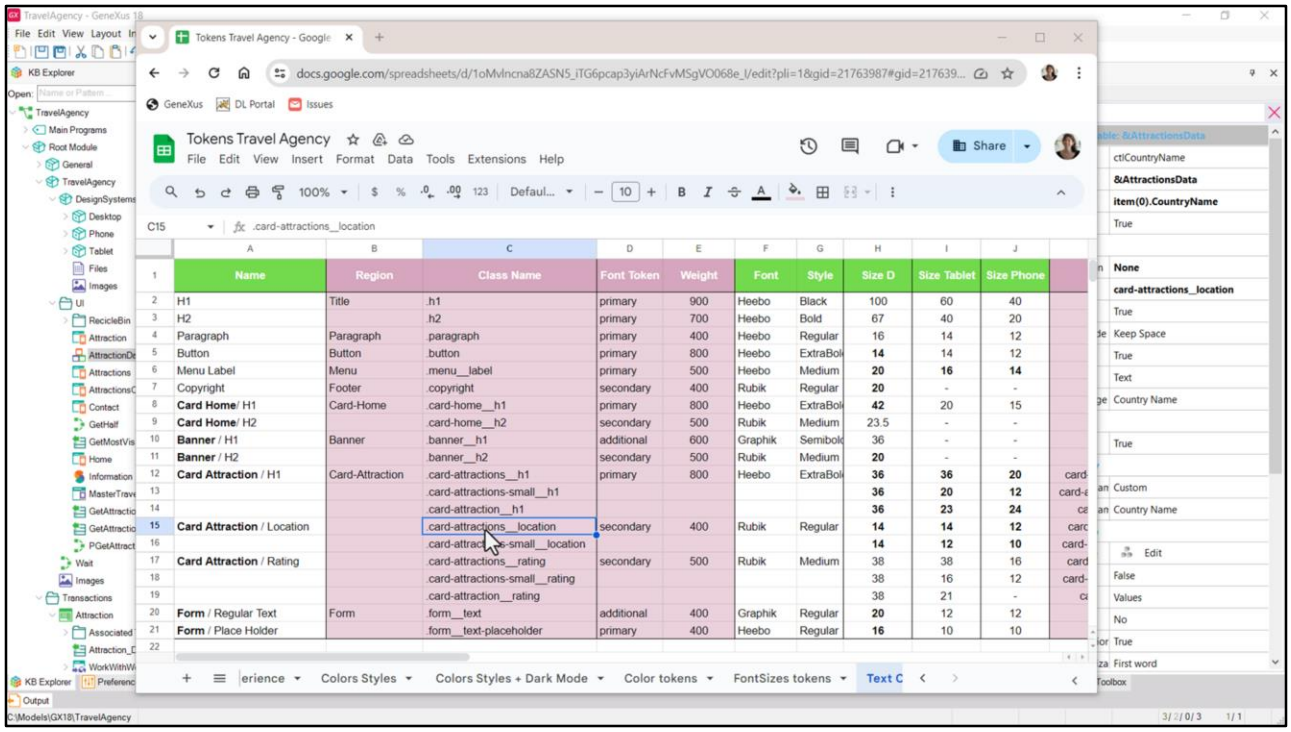
Na verdade, foi muito simples: quando fui criar o grid -farei isso com outro- e escolhi a variável SDT coleção, automaticamente me pergunta quais de seus elementos queria inserir no layout. Vejam o que acontece se eu escolher estes, por exemplo.

Aí vemos que fica igual ao grid de cima. Vou apagar este.

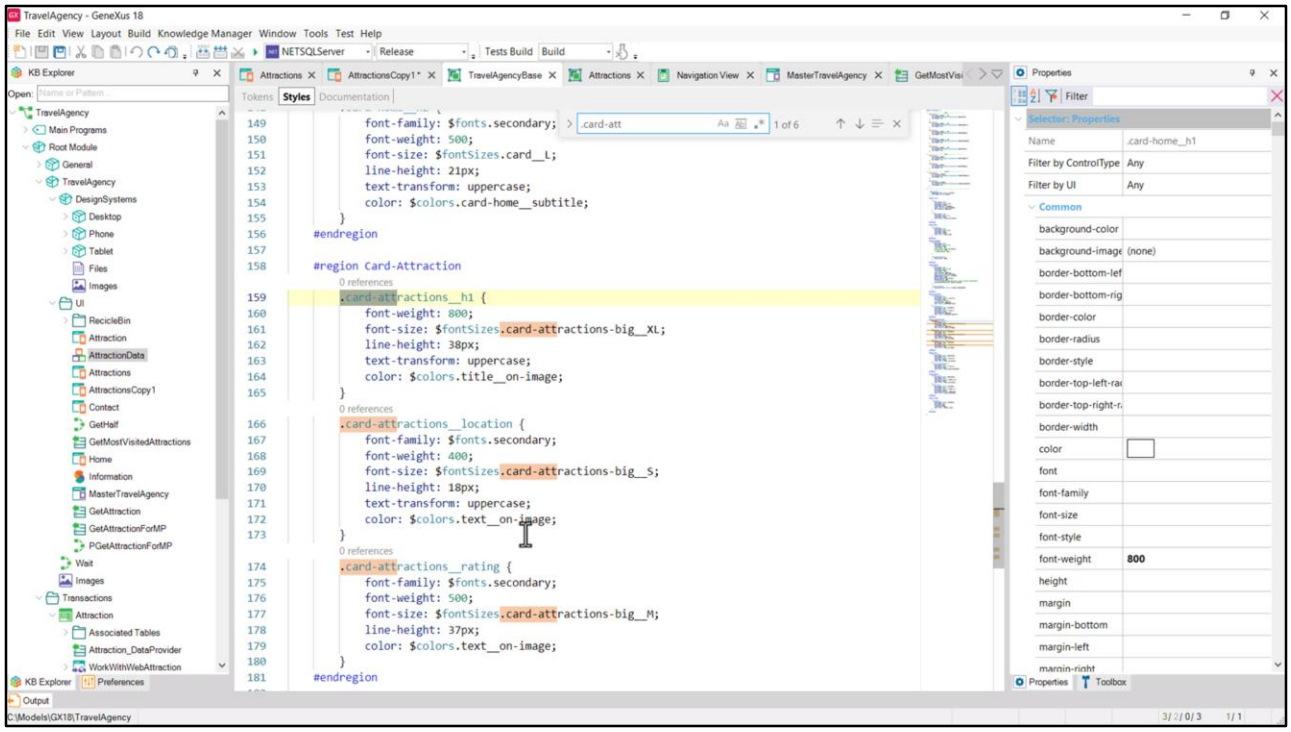


Bem, e se eu me posicionar sobre um dos elementos vemos que assim é identificado.

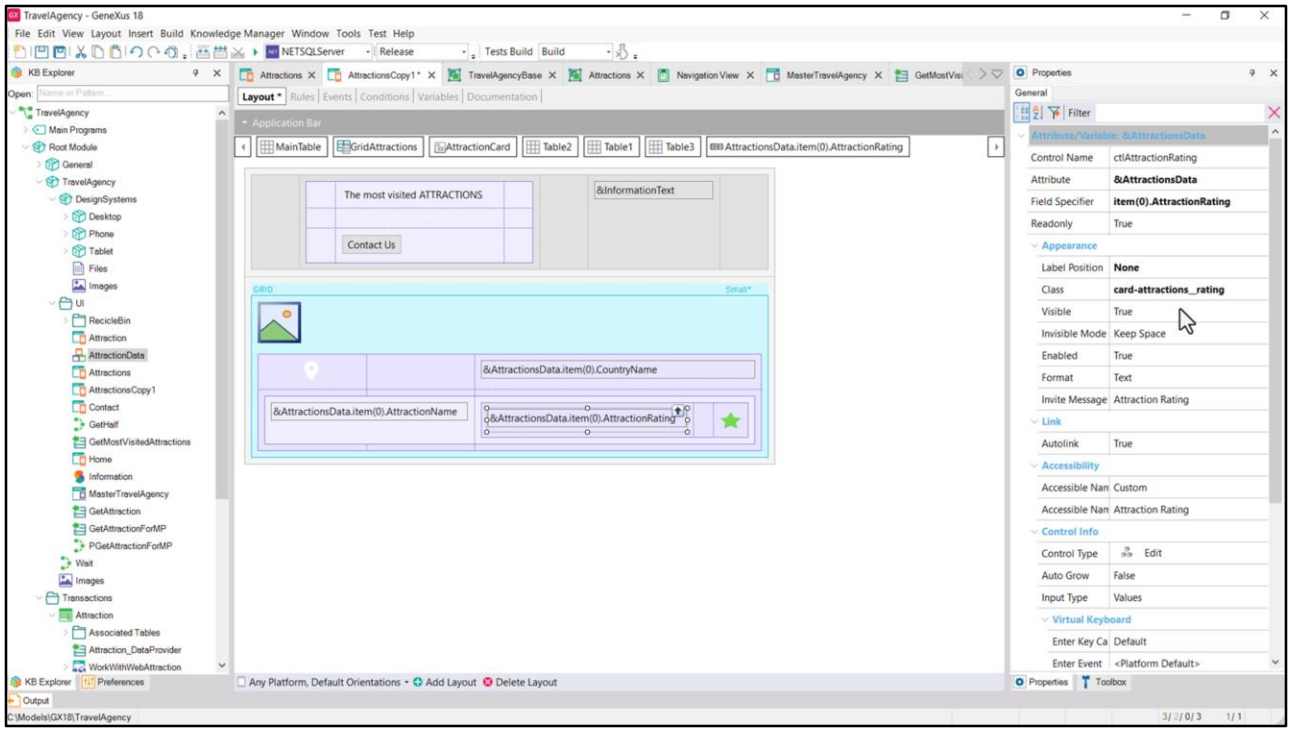
Aproveitemos para observar que deixei a Label position em None, para que não apareça o rótulo da variável que, claro, é readonly.



E também vamos ver como apliquei a todos os textos, as classes de tipografia que havíamos introduzido há um tempo, na etapa de preparação, lembram?

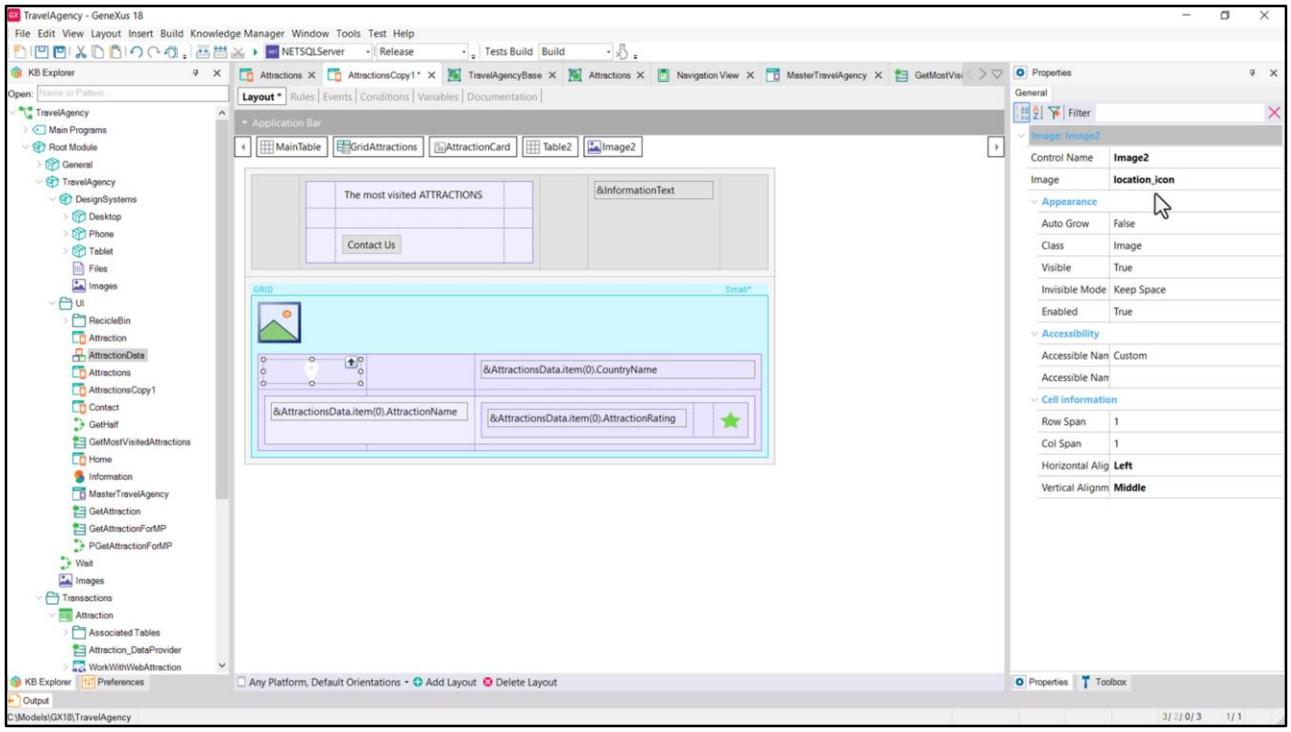


Filtro por card-attraction… e vemos na região Card Attraction essas três classes…

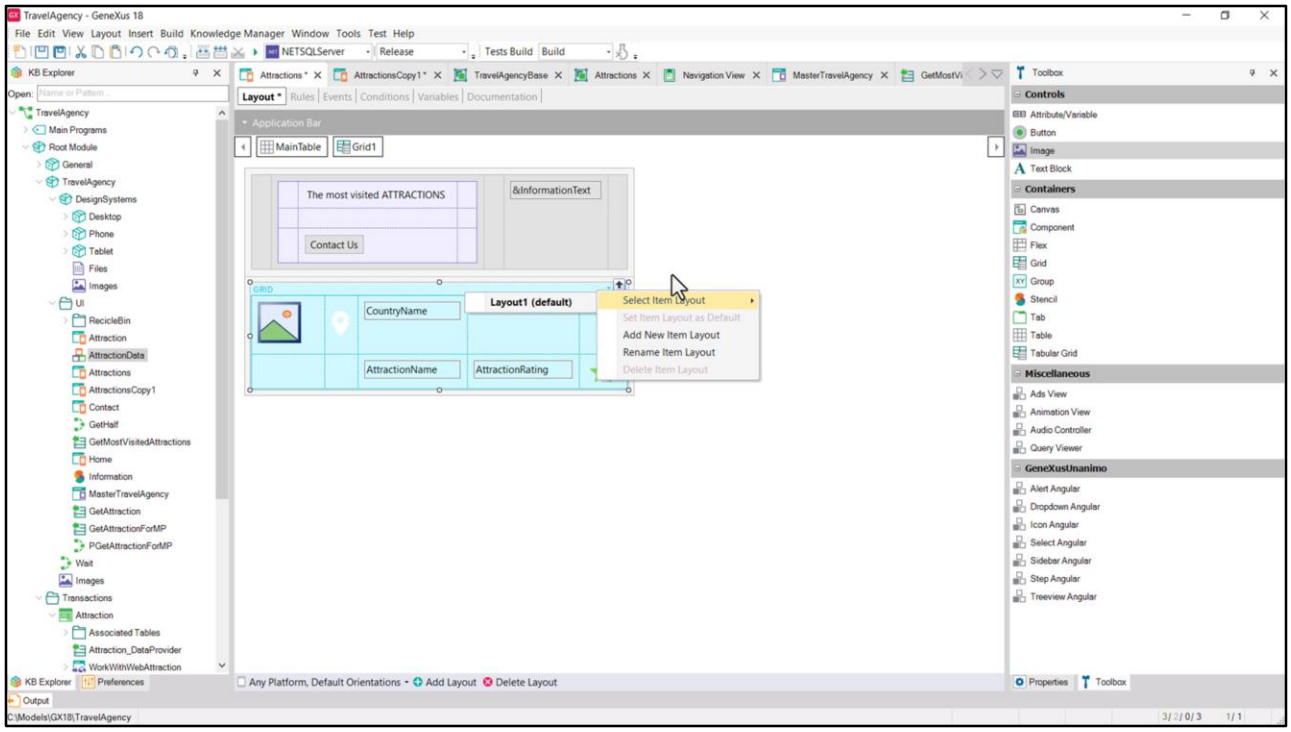


…que são justamente aquelas que aplicamos a esses elementos.

Aqui a do rating… aqui a h1.



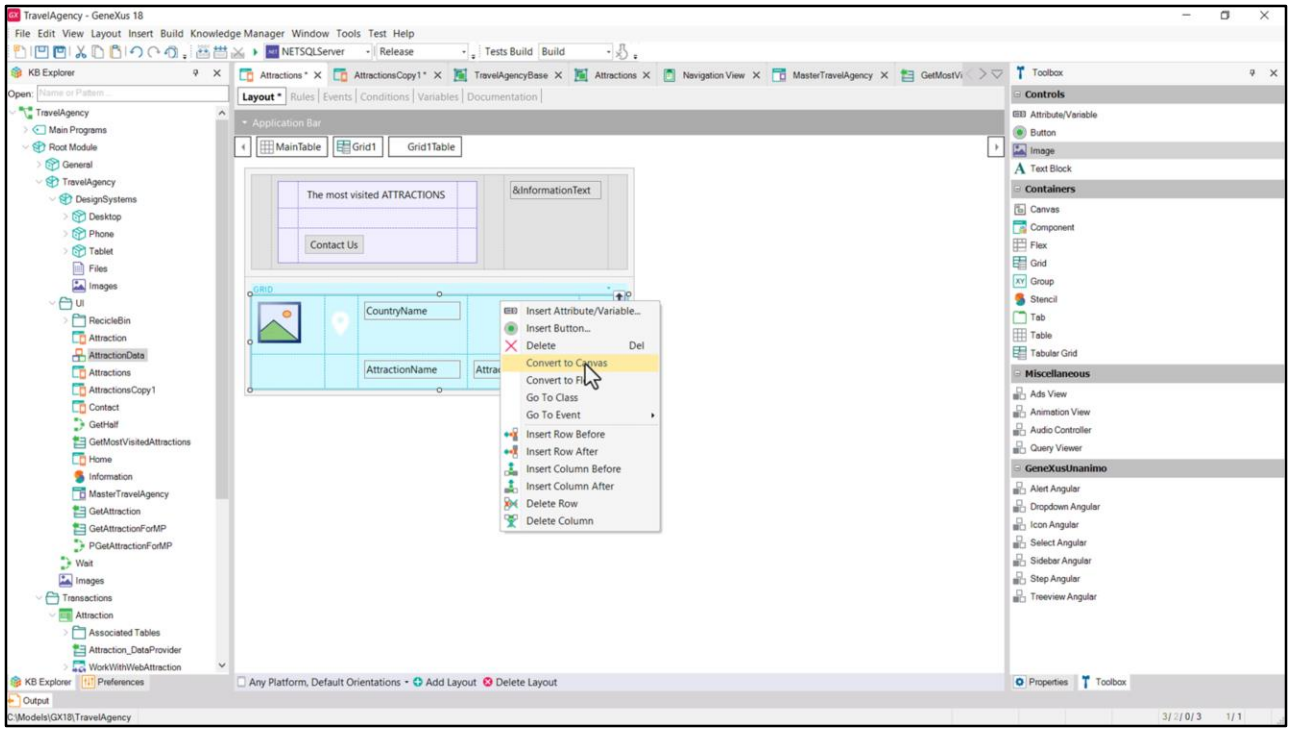
Por outro lado, tive que baixar essa imagem do Figma e inserir na KB, porque na etapa de preparação eu tinha esquecido.



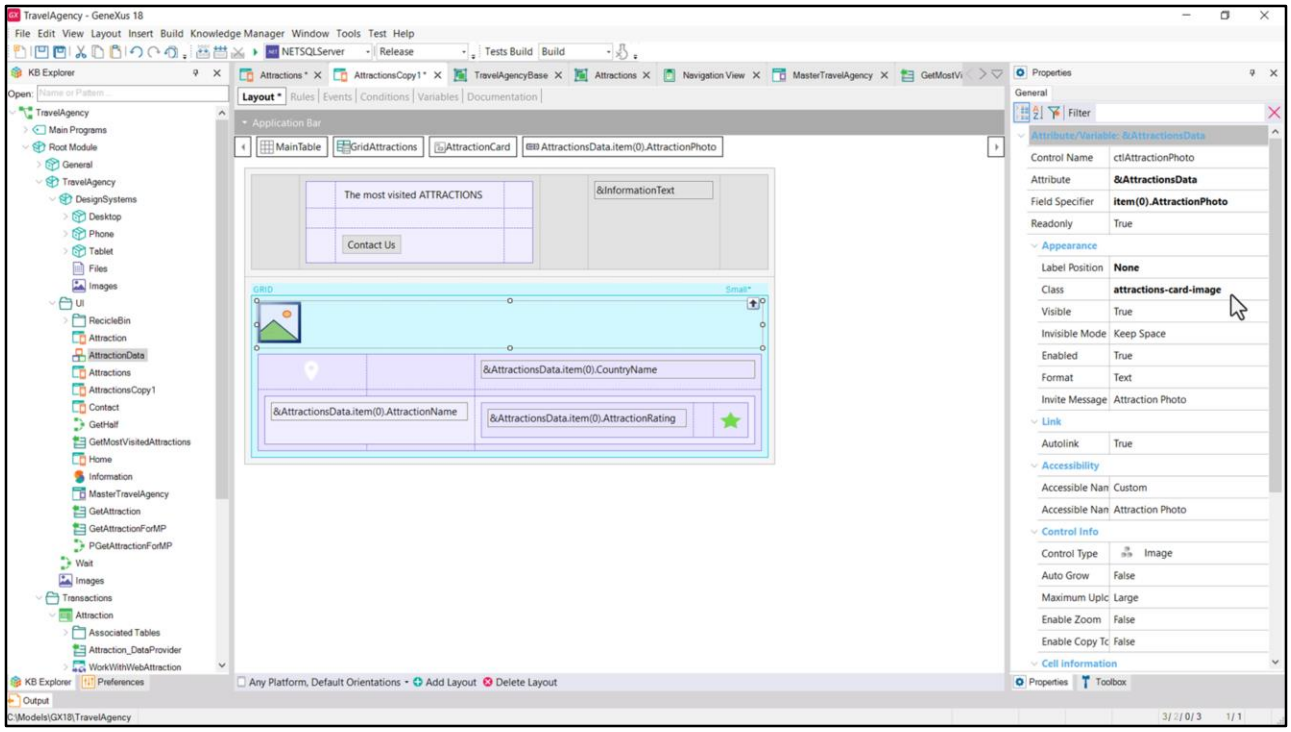
E o que eu fiz foi inserir essas duas imagens.

Podemos fazer isso em nosso outro grid, aquele que queremos implementar com atributos em vez da variável SDT, para ver, mas isso é muito simples.

E na verdade o trabalho que temos para fazer aqui, no nível do layout, que neste grid no momento é universal, é colocar todos esses controles de uma forma apropriada para poder implementar o card, onde controles serão sobrepostos...

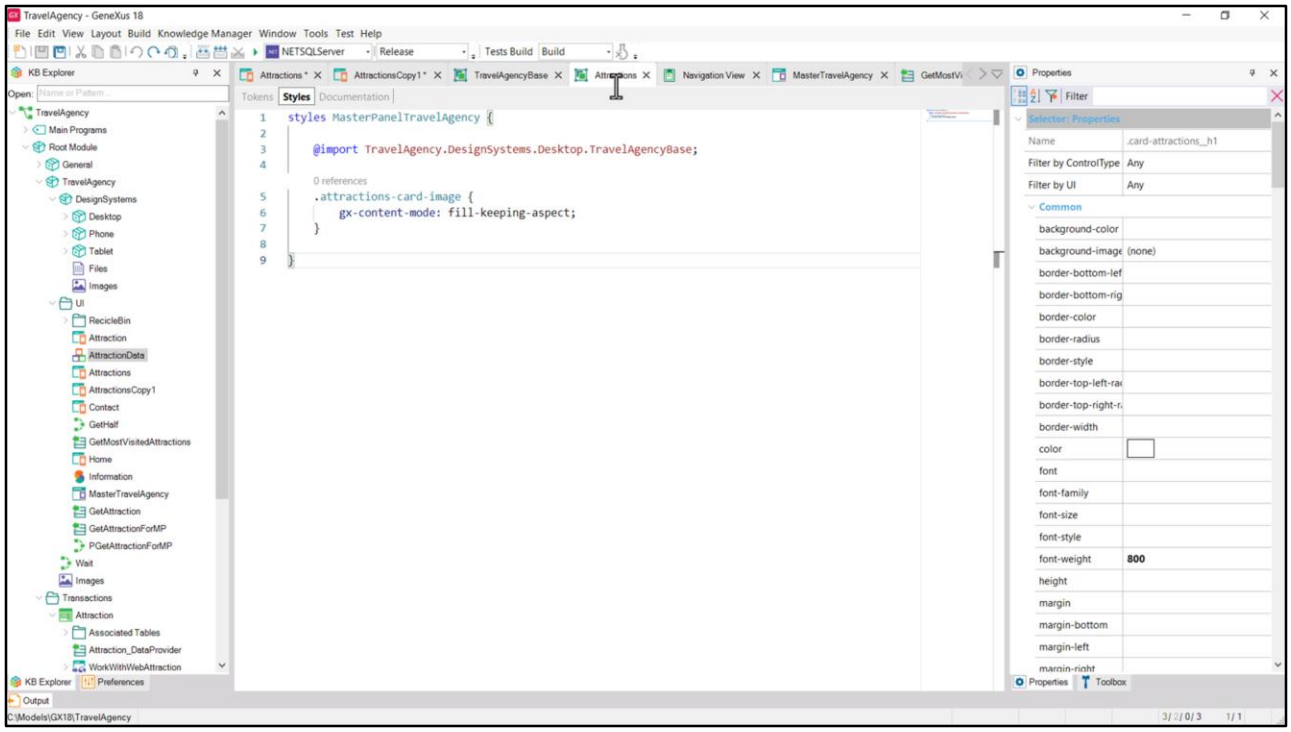


…então já sabemos que esta tabela terá que ser, na verdade, um canvas.

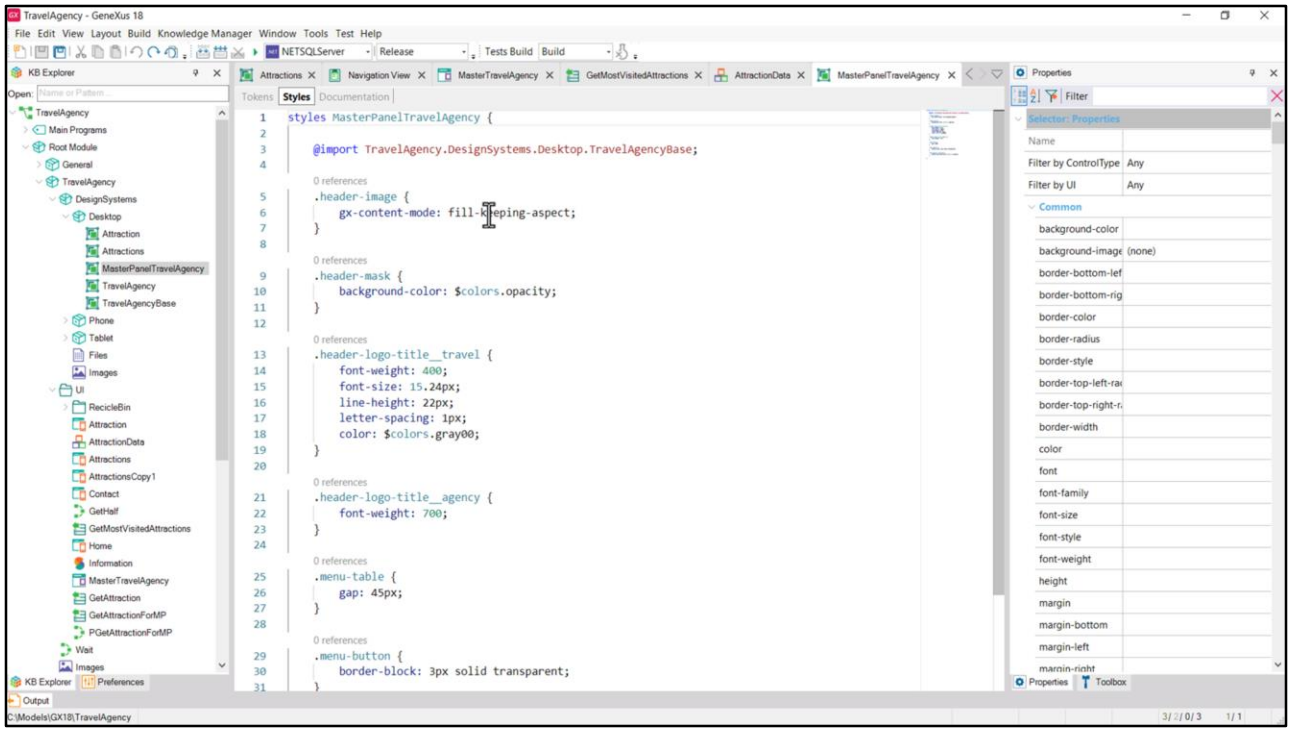


Mas vejam como eu estruturei dentro de meu outro grid, os elementos internos do canvas.

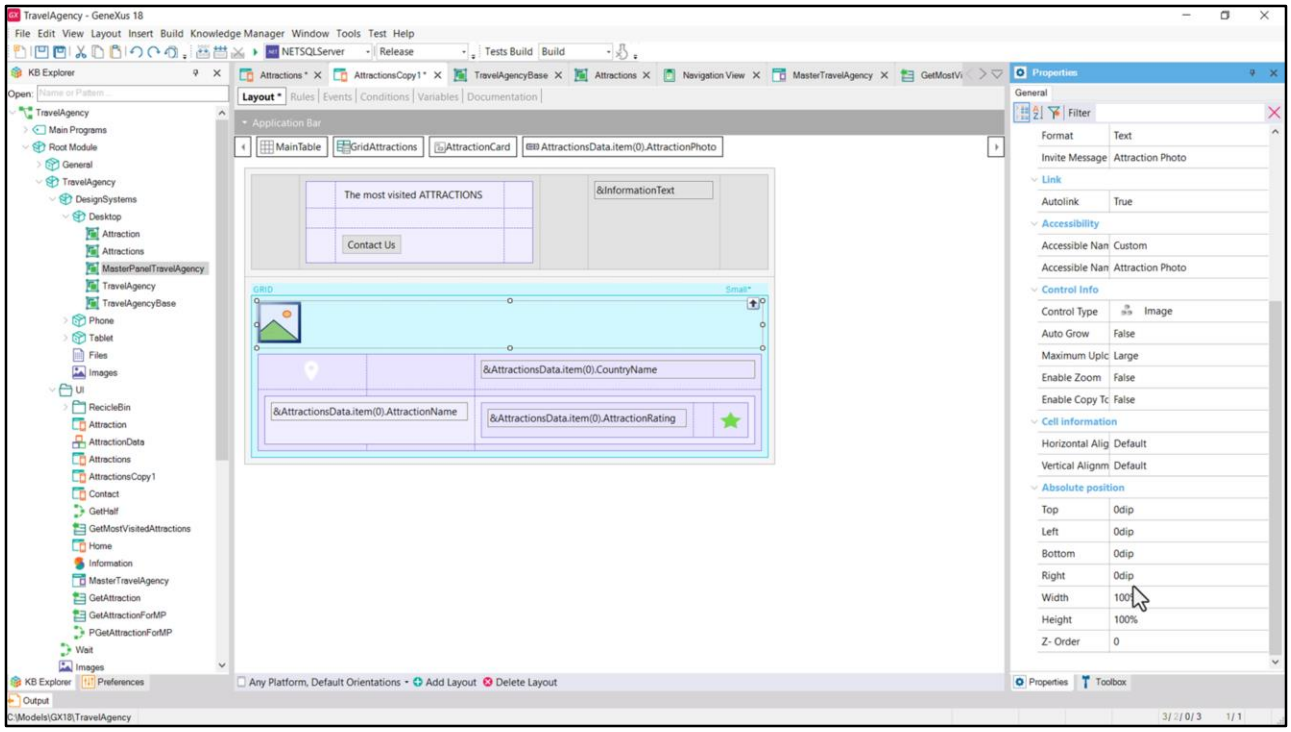
A imagem, solta, vejam que tem essa classe que criei especialmente dentro de um DSO específico para desenhar as especificidades desse panel.



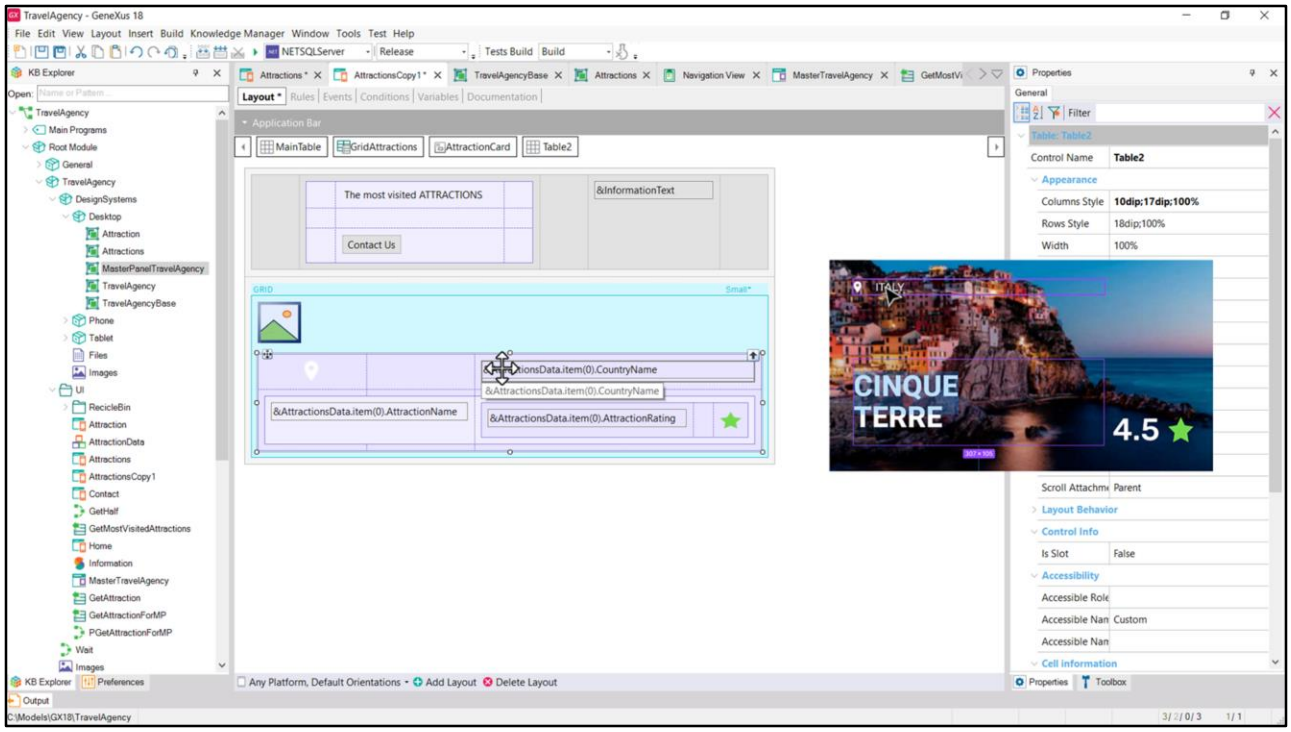
É por isso que chamei do mesmo nome do panel: Attractions. Aqui tenho a classe.



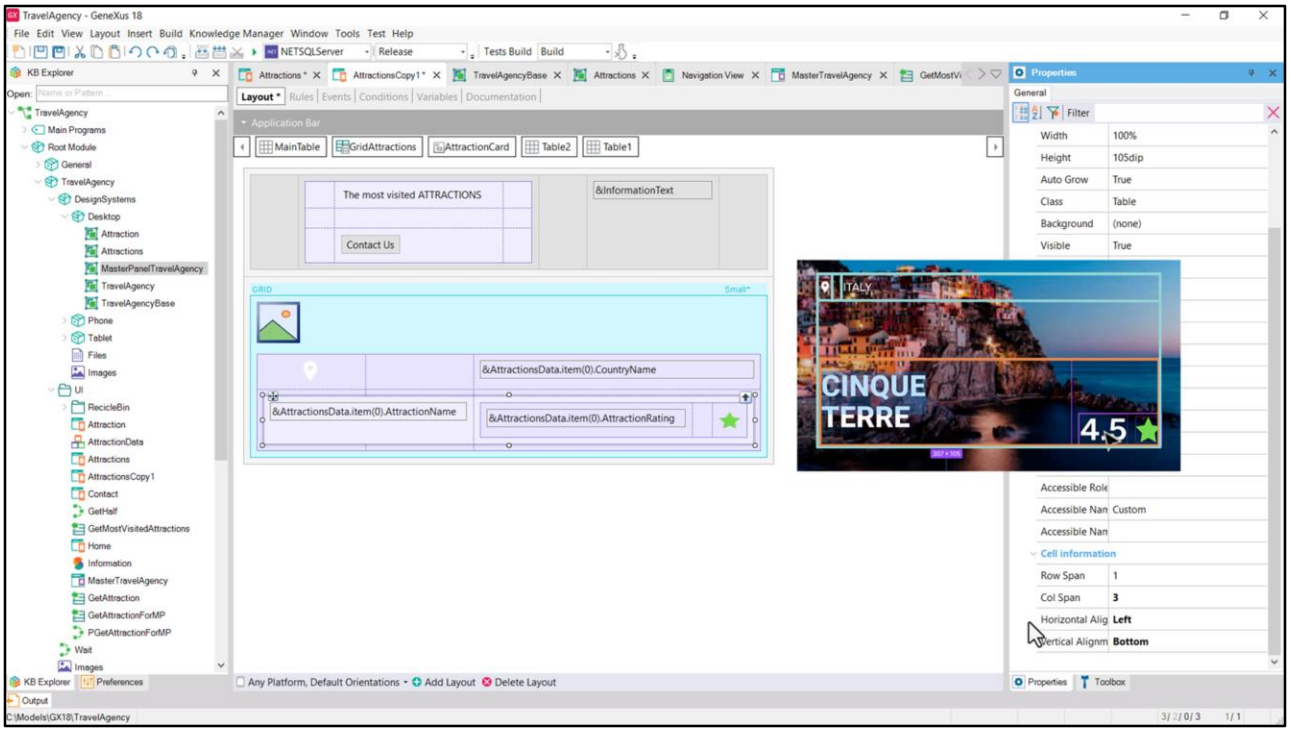
Já fizemos isso antes, lembram? Para o Header. Mas não quis utilizar aqui sua mesma classe, caso eu precisasse diferenciá-las mais tarde.



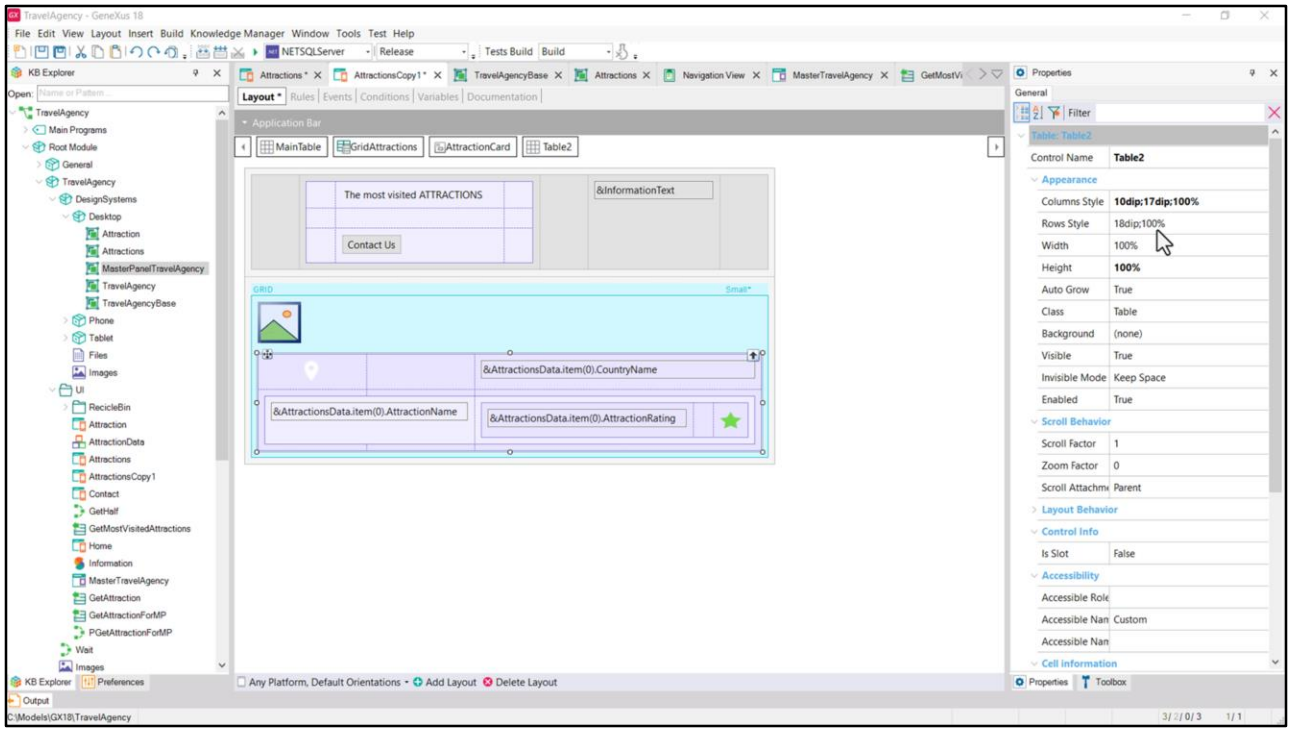
Devemos dar seu posicionamento absoluto e como vemos, será tal que ocupe 100% da largura e altura do canvas, colando às suas bordas. E a camada, a mais profunda.



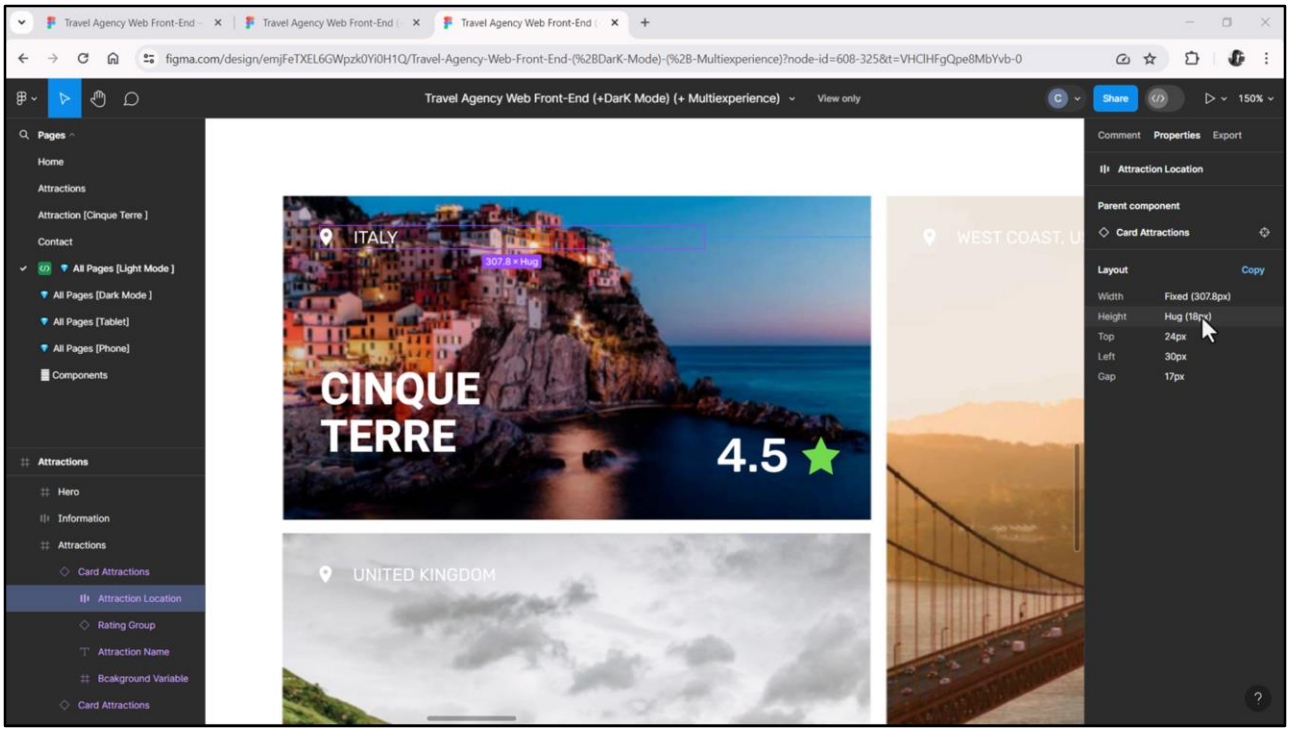
Então vemos que coloquei todos os outros elementos em outra tabela, com duas linhas e 3 colunas. Na primeira linha coloquei o ícone, um espaço e o nome do país…



E na segunda linha outra tabela, que se expande entre as 3 colunas da tabela container. Alinha-se à esquerda horizontalmente, e verticalmente abaixo, em relação à célula que a contém, que corresponde à segunda linha desta tabela.

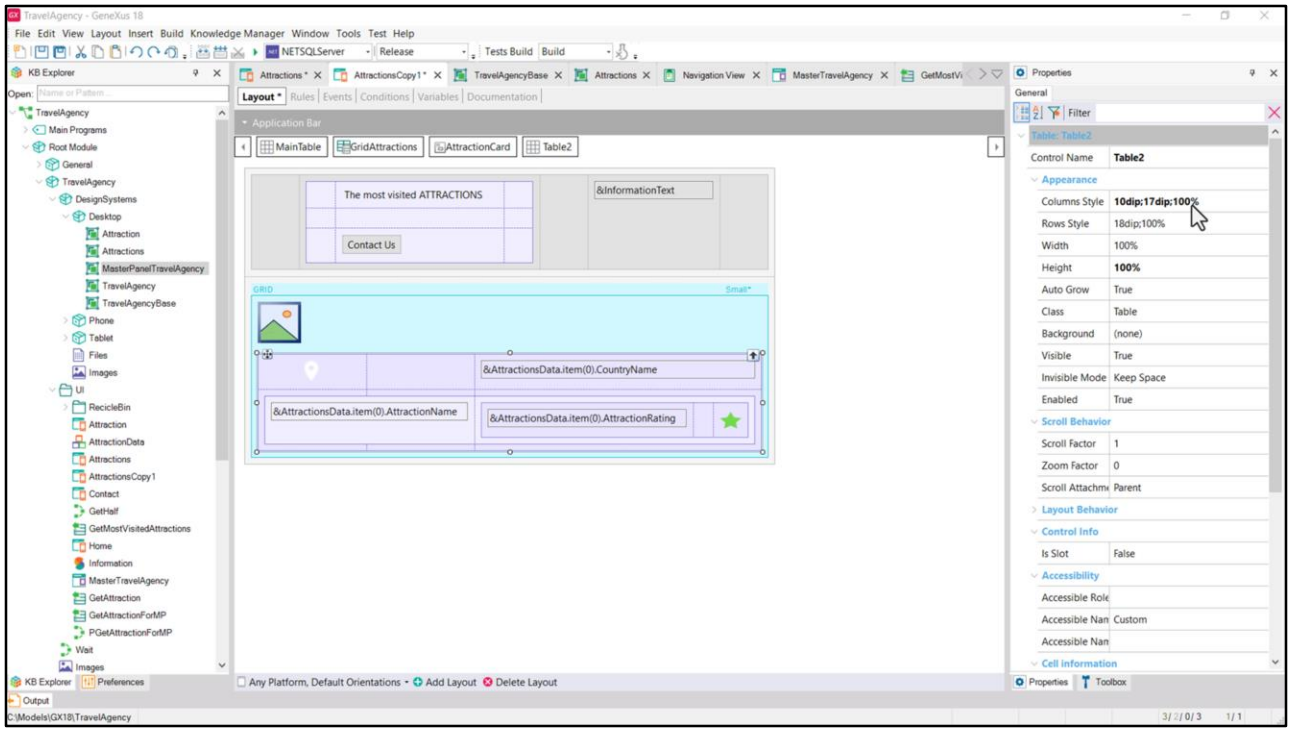


À qual vemos que dei de altura 100%, porque para a primeira linha coloquei 18 dips…

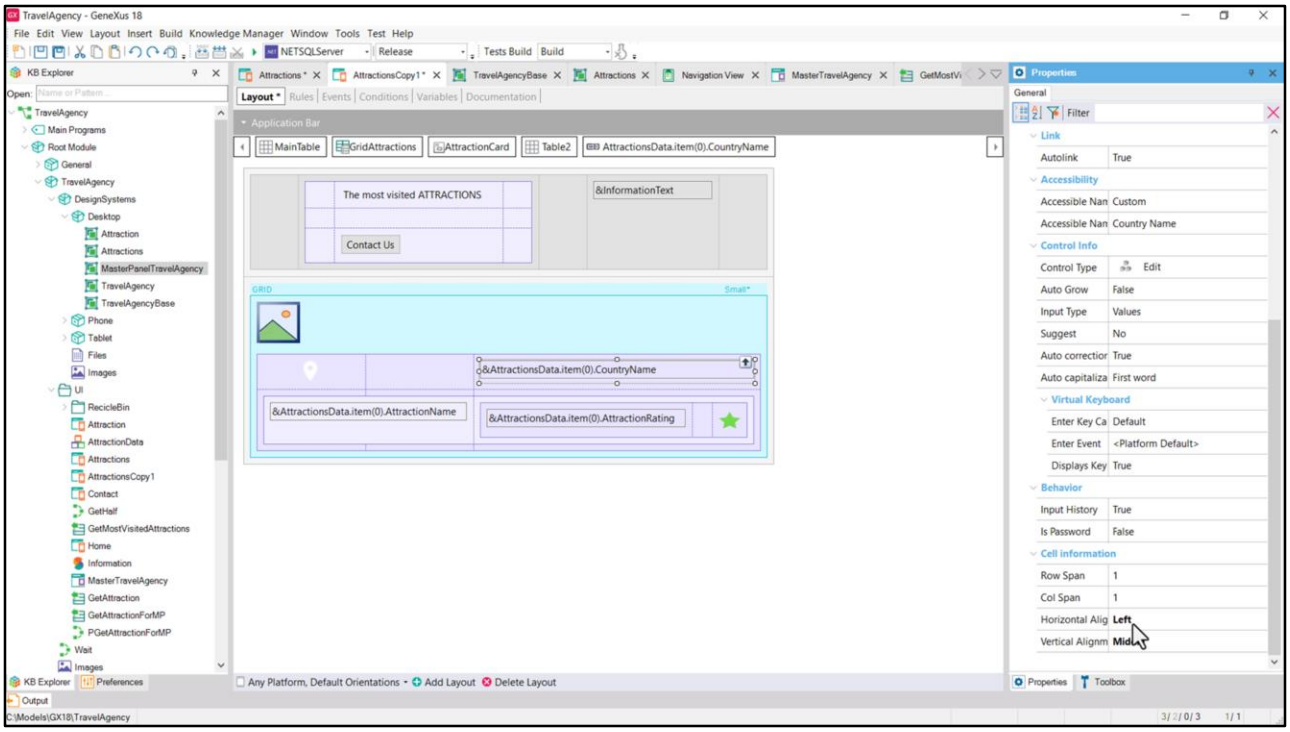


...que extraí daqui.

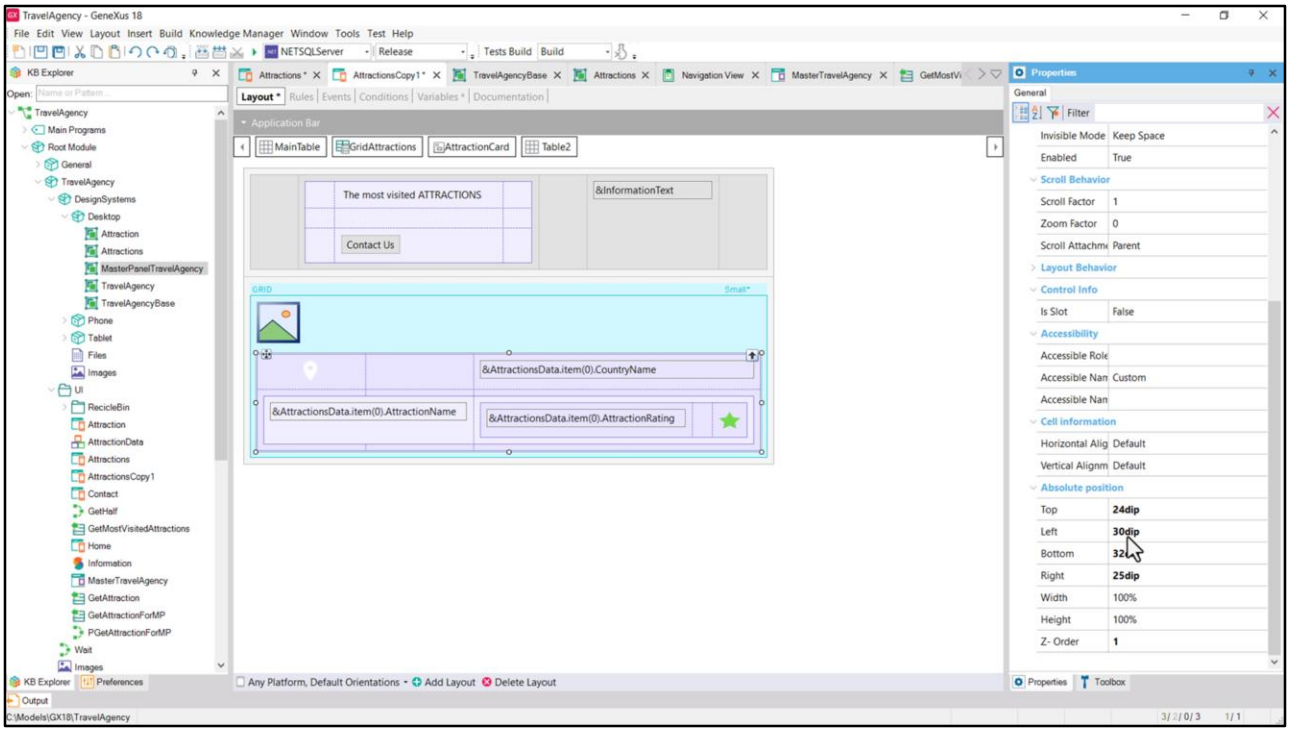
E já estamos vendo esses 24 dips de cima do canvas, esses 30 da esquerda e este gap de 17 dips entre o ícone, de quase 10 dips de largura, e o texto ITALY. E que estão centralizados verticalmente, também podemos ver.



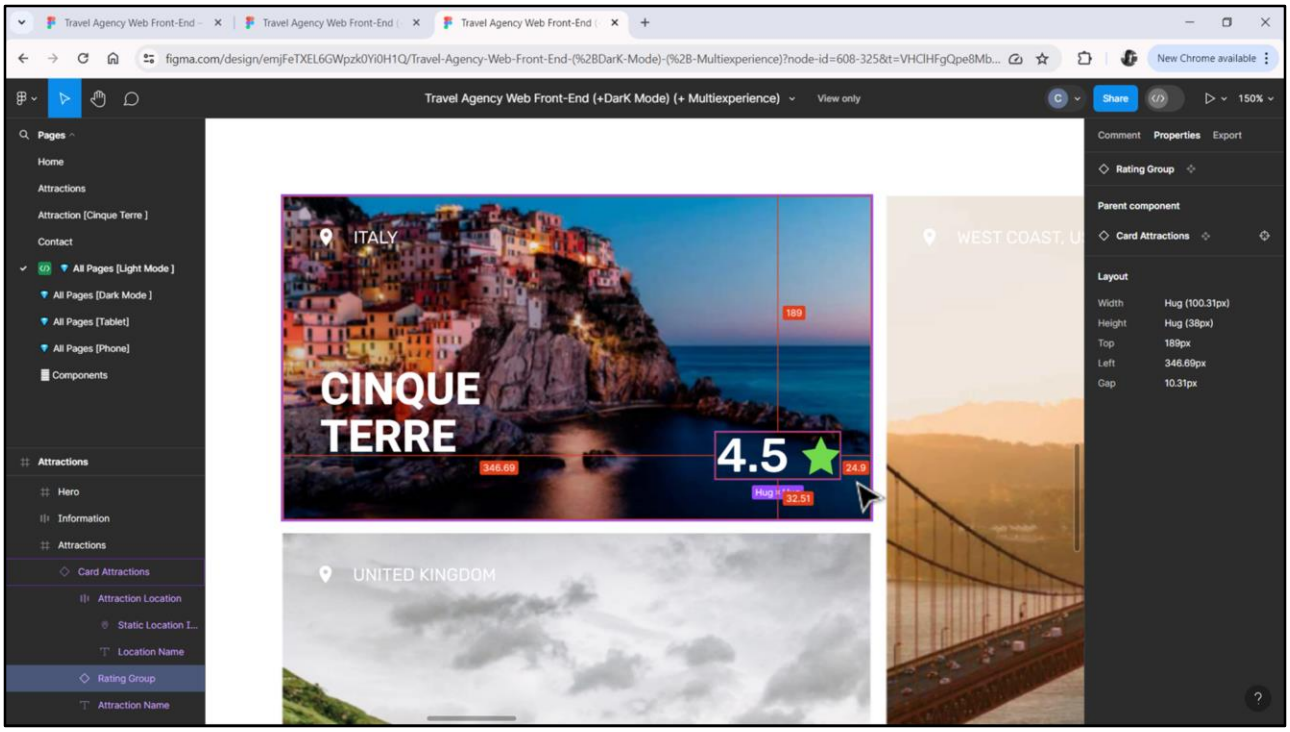
E é assim que deixei 10 dips para a coluna do ícone, 17 de espaço e a terceira coluna para expandir para os 100% restantes.



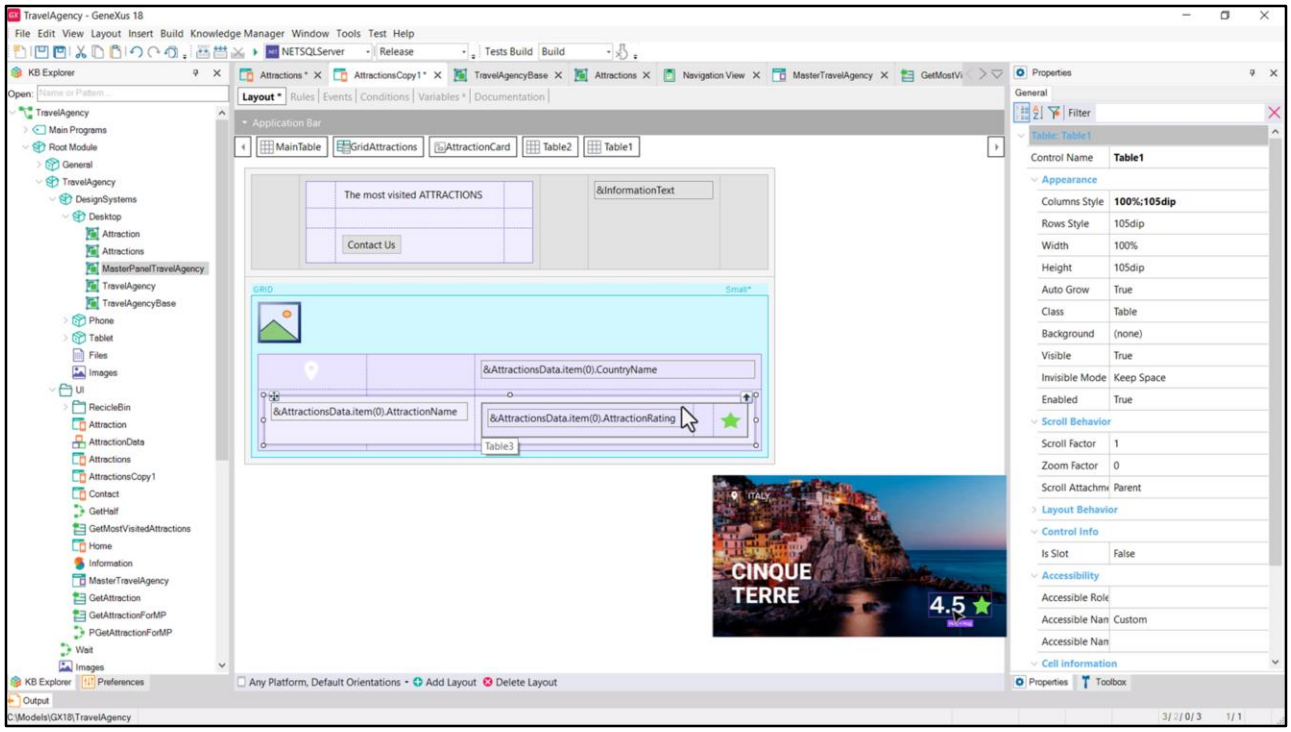
Mas com a condição de que essa variável esteja horizontalmente alinhada à esquerda.
E ambos os controles, verticalmente, no meio.



A tabela está dentro do canvas, então foi posicionada em relação às suas bordas… Acabamos de ver que começava a 24 dips de cima e a 30 da esquerda. E vamos ver de onde vem esses 32 de bottom e esses 25 da direita.

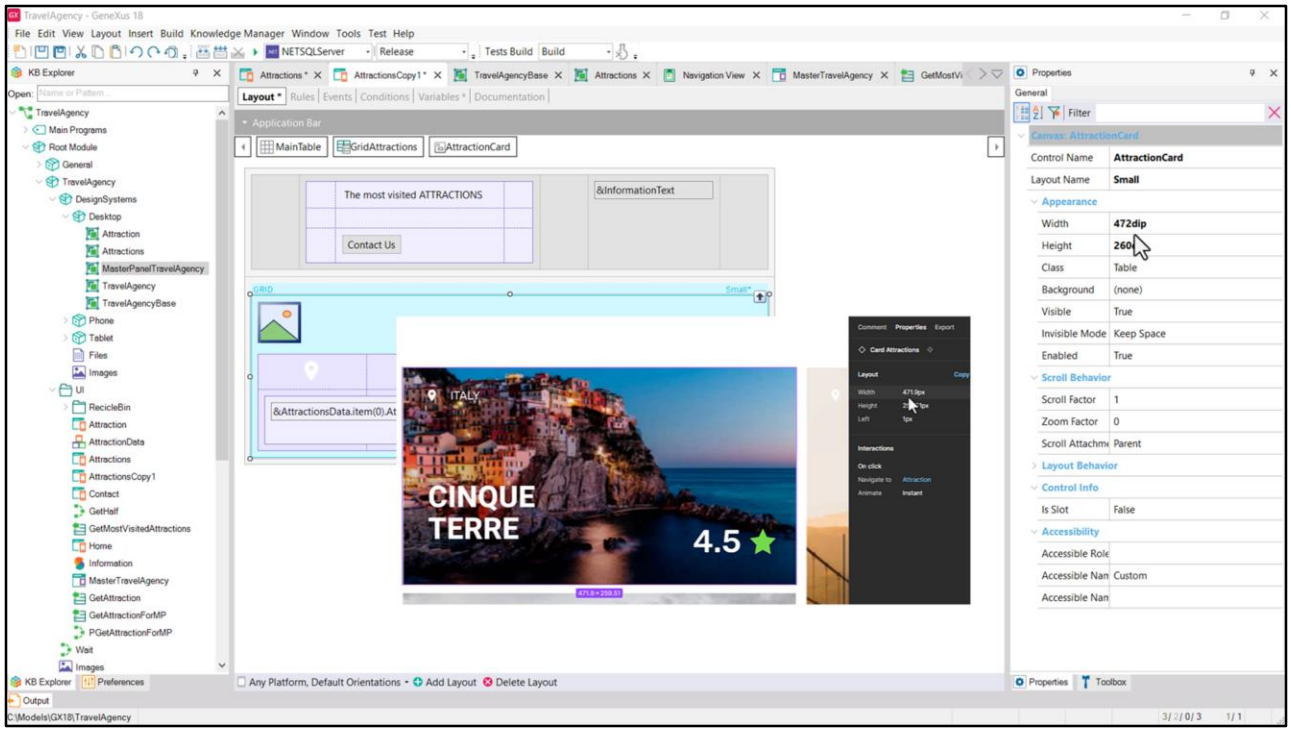


Aí os vemos. Nós os arredondamos.



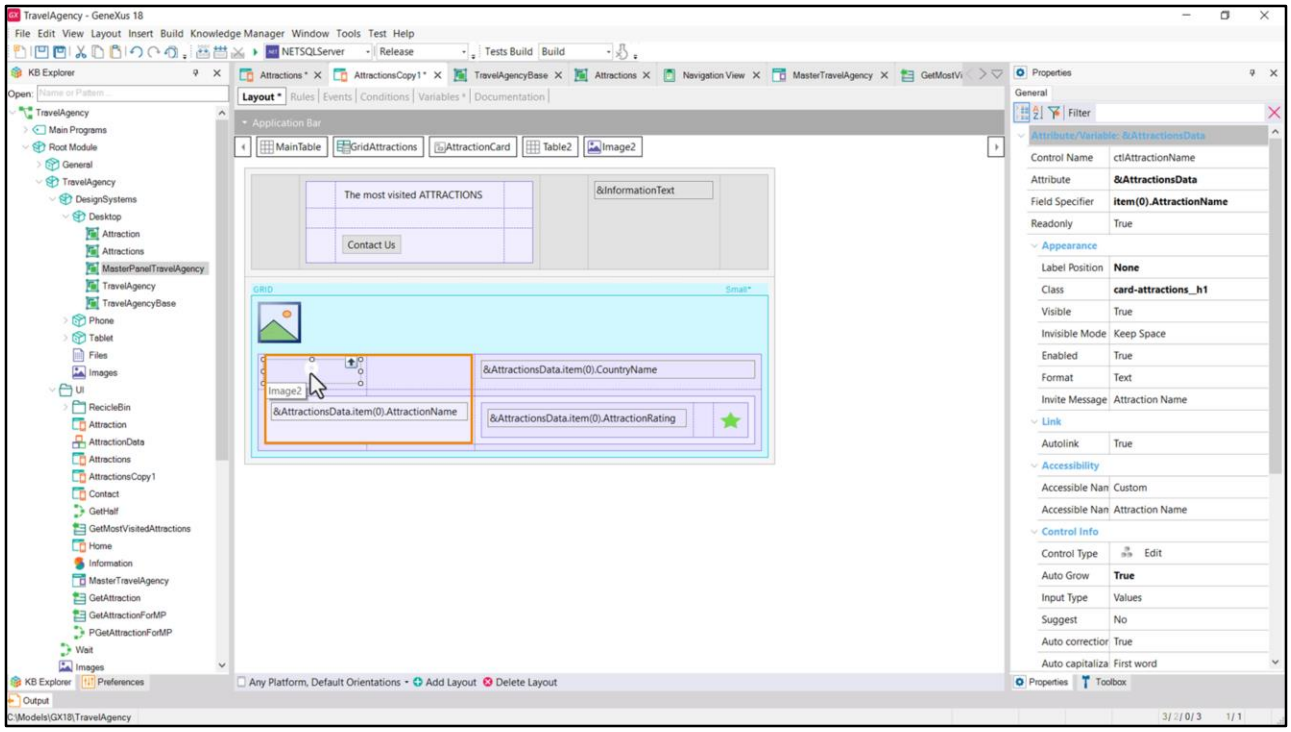
Bem. Colocamos a tabela com Z order 1, para que ela fique por cima da imagem.

E agora vamos analisar esta subtabela. Vejam que ela tem 2 colunas, a da direita de 105 dips, que é o espaço máximo que poderá ocupar, de largura, essa outra tabela; e a da esquerda, que conterá essa variável, que tem um espaço de 100% restante da largura dessa tabela para ocupar.



E qual é a largura desta tabela? 100% de seu container, que é a linha 2 inteira, que por somar as 3 colunas porque se expande nas 3 colunas, é claramente 100% da largura da tabela….

Que é qual, então? O que resulta da subtração da largura do canvas… as bordas Left e Right.

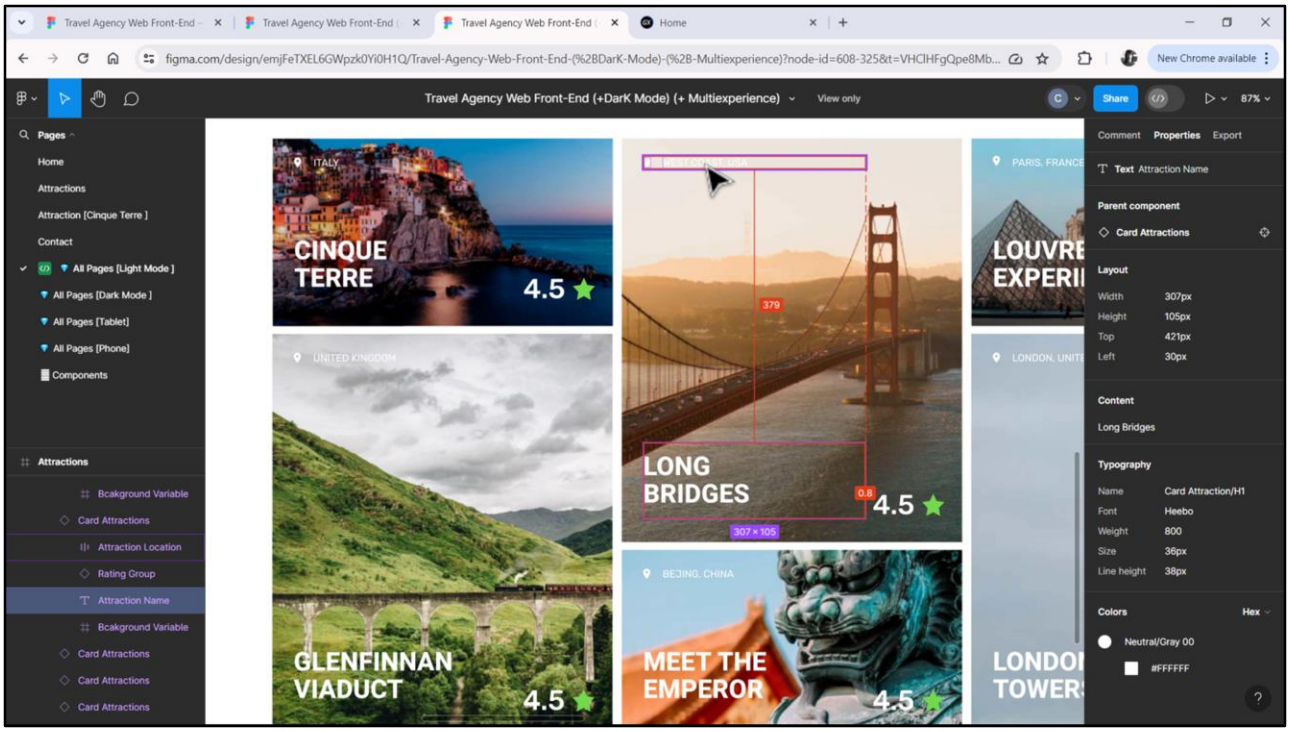


Eu poderia continuar analisando cada valor que dei a cada propriedade, mas como é mais do mesmo e não quero entediá-los, vamos parar por aqui. No xpz que deixo, podem investigar tudo isso em detalhes.

Como eu venho repetindo desde o início, quase nunca há apenas uma maneira de implementar as coisas. A vantagem de ter colocado este e este controle em uma tabela é que modelo o alinhamento pela esquerda de forma ideal. Imaginem que é preciso alterar a distância da borda esquerda do Canvas, por exemplo. Faço isso apenas uma vez, para a tabela. Se eu tivesse esses controles soltos ou em tabelas independentes, eu teria que fazer a alteração para cada um.

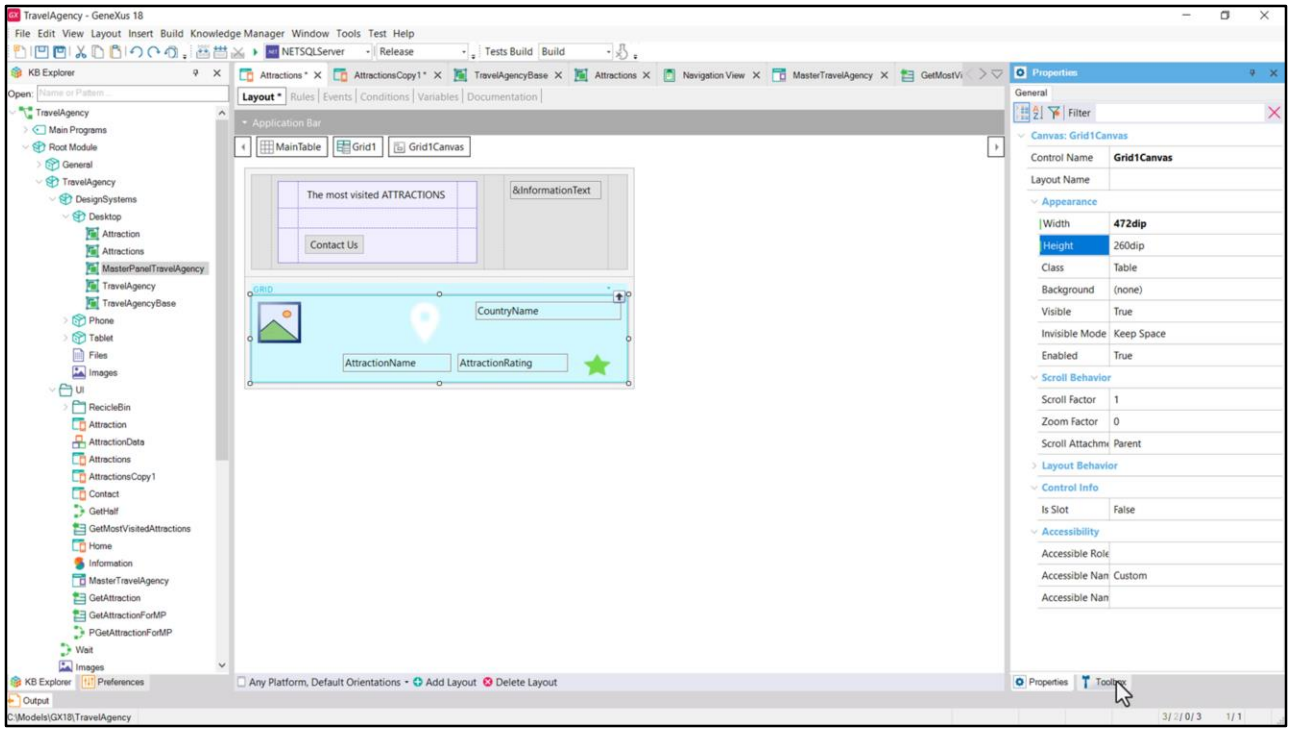
O mesmo vale para controles que sabemos que precisam ser alinhados horizontalmente de algum modo, como estes.

Quando há no design uma estrutura óbvia que inter-relaciona os controles, então convém utilizar tabelas (nas variedades apropriadas, como a flex para alguns casos, é claro).



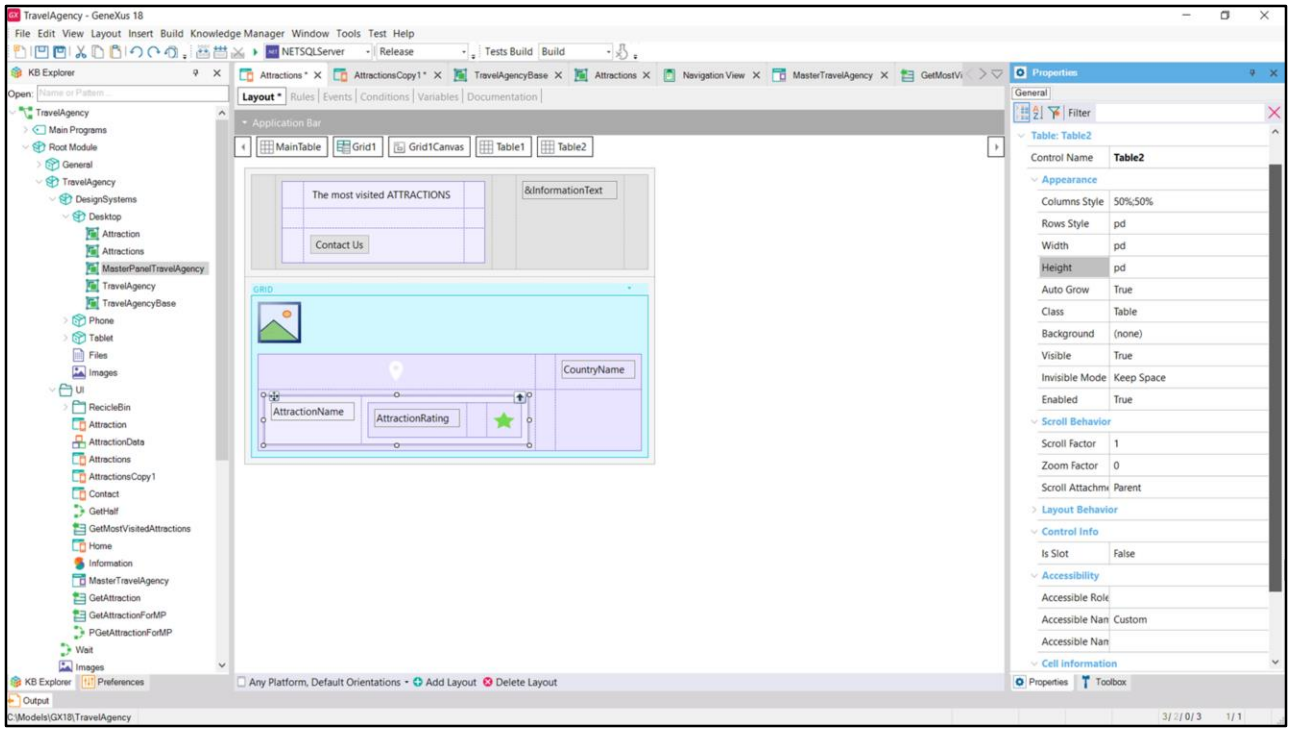
Agora quero contar como analisei as diferenças entre os cards pequenos e os grandes.
Um, sabíamos que tinha 260 dips de altura e o outro, 560. Essa diferença de 300 dips se deve unicamente a esse espaço vertical.

Ou seja, para converter um card grande em um pequeno, basta remover esse espaço do meio de 300 dips.
Observem que a distância entre este elemento e este outro é de 80 dips aqui, enquanto aqui é de quase 380. Ou seja, 300 é a diferença entre uma e outra.



Então, pude primeiro me dedicar a modelar o card Small como se fosse o default. E quando terminei, foi quando implementei o Large. E poderíamos fazer a mesma coisa, muito rapidamente, com nosso grid com atributos.

Aqui defino as dimensões do canvas.

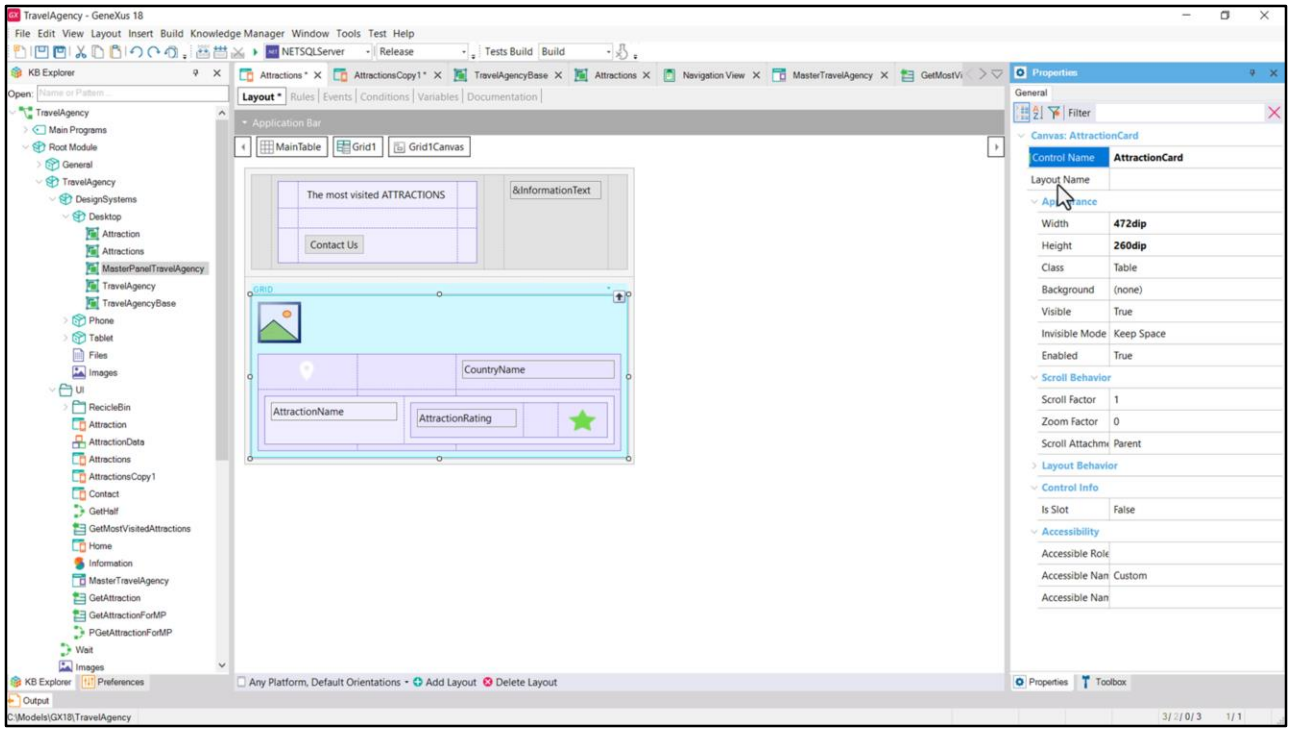


E então começo a modelar os controles tabela, a colocar nossos controles imagem e atributos dentro das tabelas que identificamos.

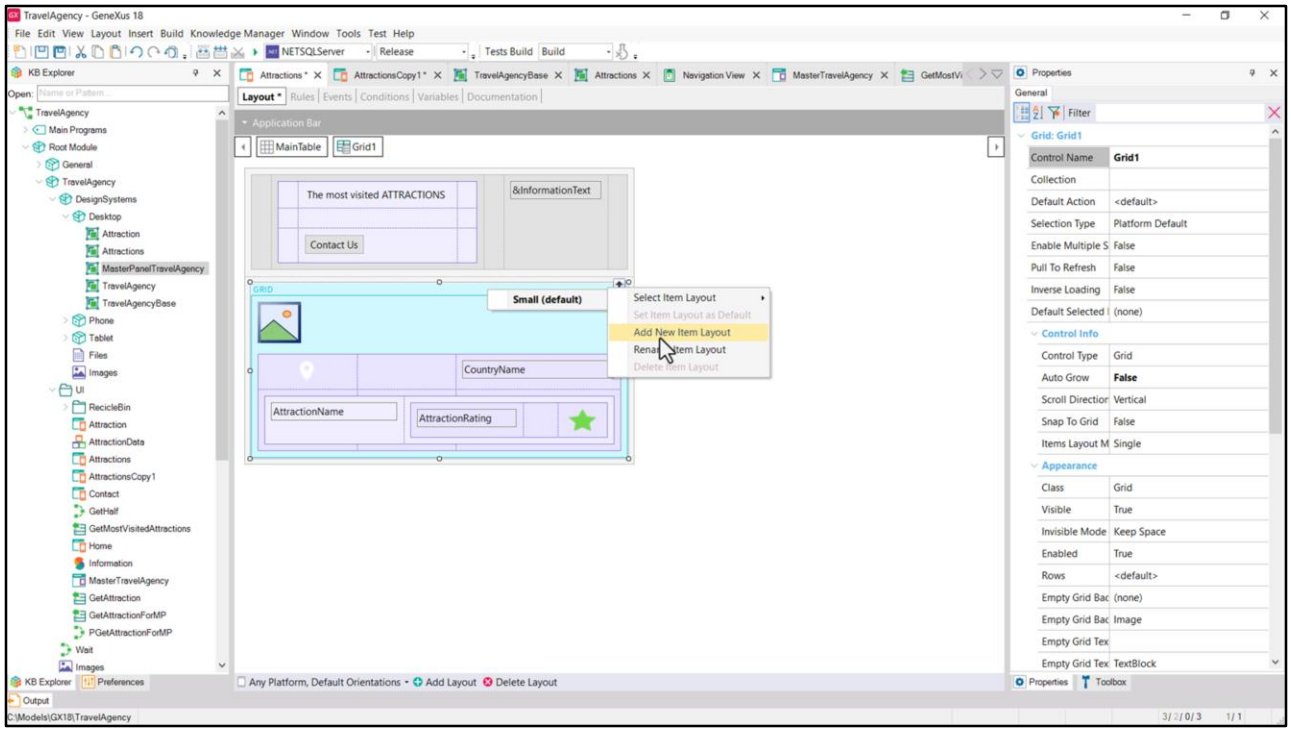
Por último, coloco esta, como segunda linha da tabela principal e faço com que se expanda nas 3 colunas.

Bem, o restante é basicamente copiar as propriedades que tínhamos em AttractionsCopy para este outro.

Começo a fazer isso rapidamente, mas deixo como tarefa para vocês, para não ficarmos entediados. Na verdade, teria sido melhor copiarmos o canvas inteiro e ali substituir as variáveis pelos atributos.

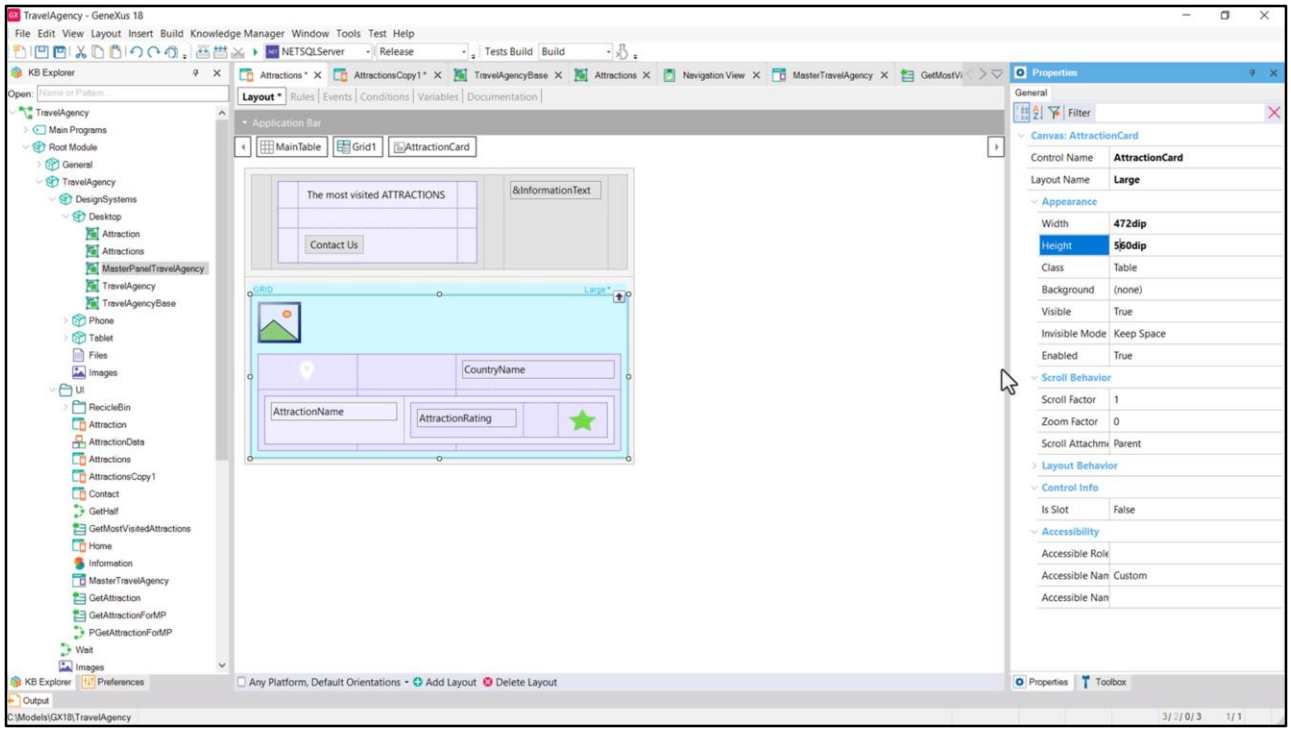


Bom, aqui já temos o card pequeno pronto. Vou alterar o nome do canvas para que se chame AttractionCard, como no outro grid.



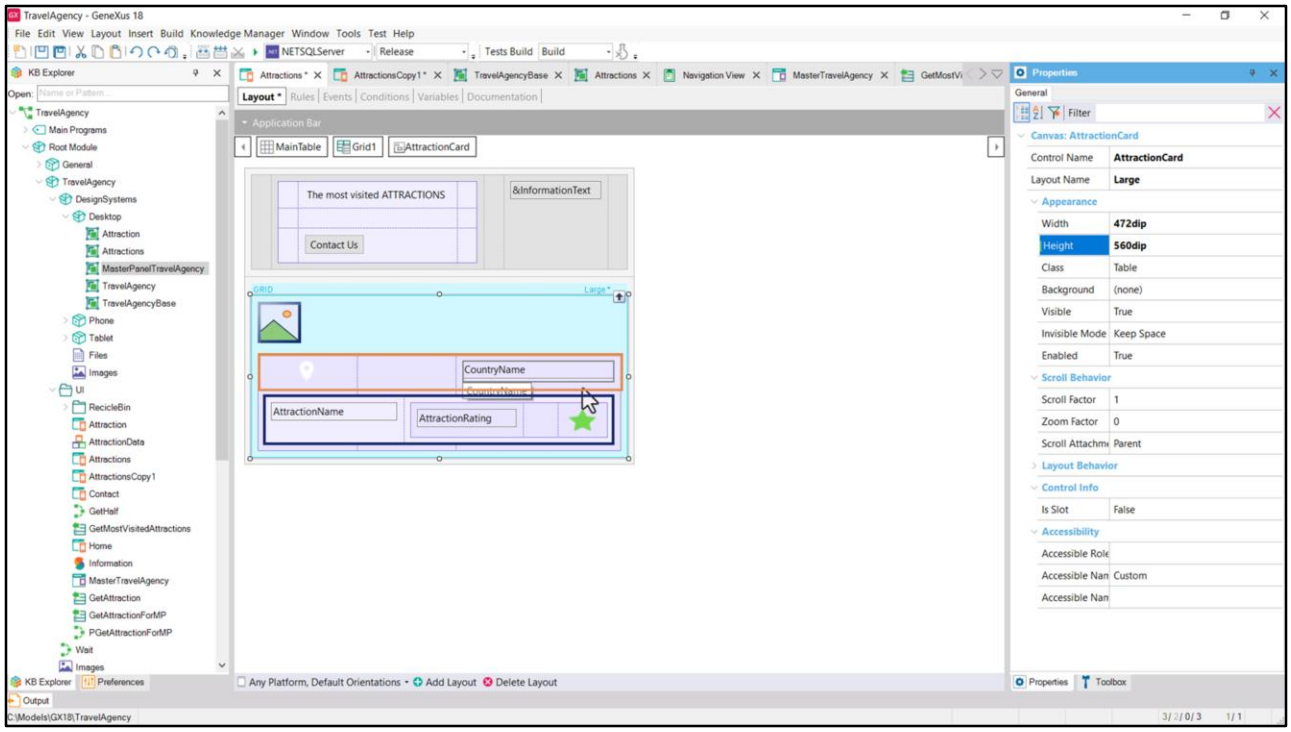
E vou mudar o nome desse layout do item para Small. Será o default.

E agora vou adicionar outro item layout, que chamarei de Large…



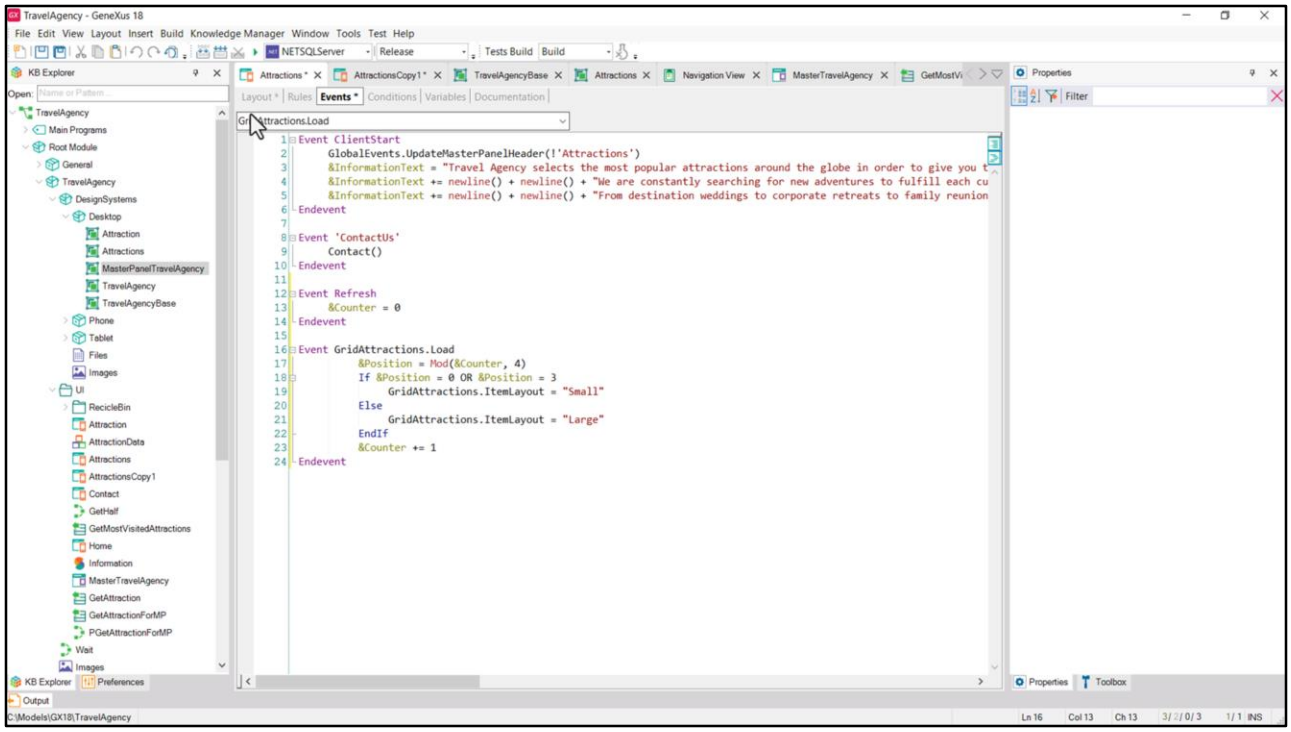
É inicializado com o layout do único que existia. Assim, terá exatamente os mesmos controles, com os mesmos valores para as propriedades.

Como já vimos, a única diferença entre o layout Small e o Large será na Height... aqui será de 560 dips.



Vamos analisar por que, apenas fazendo isso, já haverá mais 300 dips entre a linha desta tabela e a tabela da linha 2.

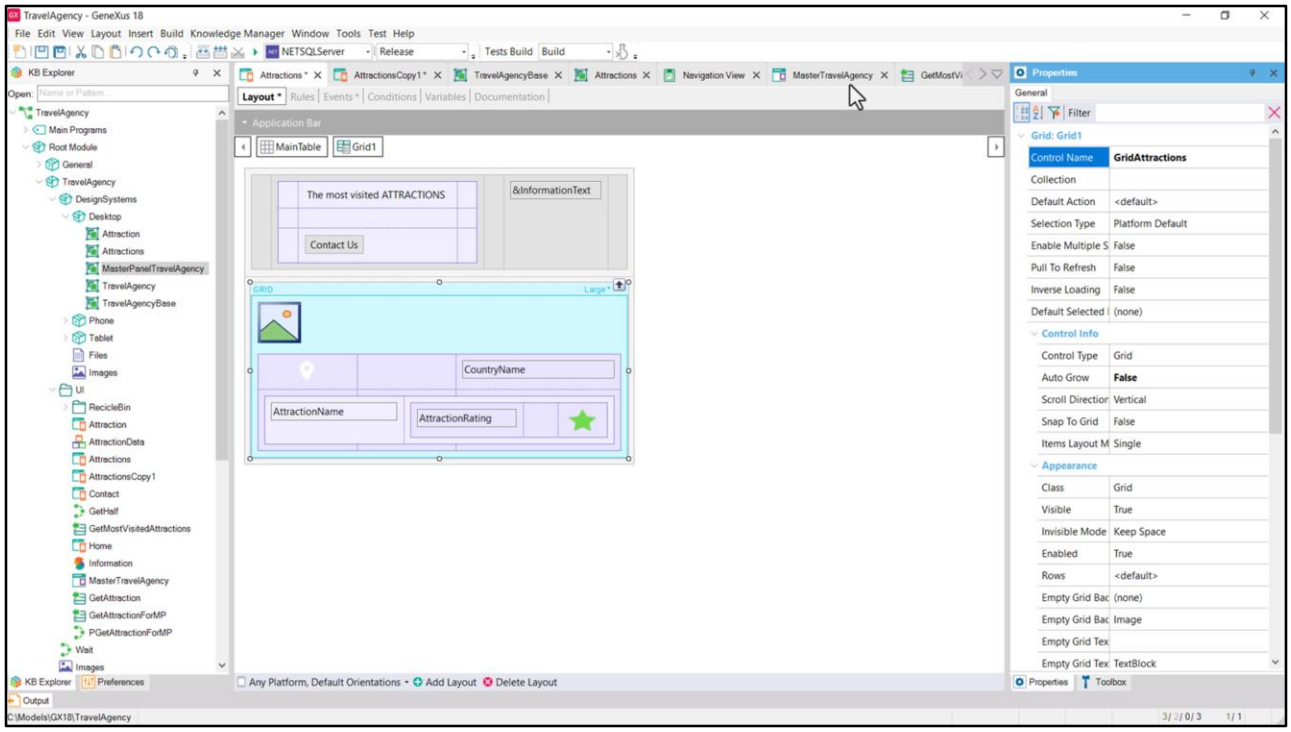
A tabela container tem fixadas suas bordas nessas distâncias: de cima e de baixo. A altura fica relativa à altura do canvas, que é o que varia entre ambos os layouts. Mas além disso, esta tabela tem uma altura fixa e está alinhada verticalmente abaixo, por isso ficará sempre sobre a borda inferior. E, portanto, o espaço que sobra para completar os 100% da altura da linha onde está localizada a tabela é o que produzirá essa diferença de 300 dips entre um caso e o outro.



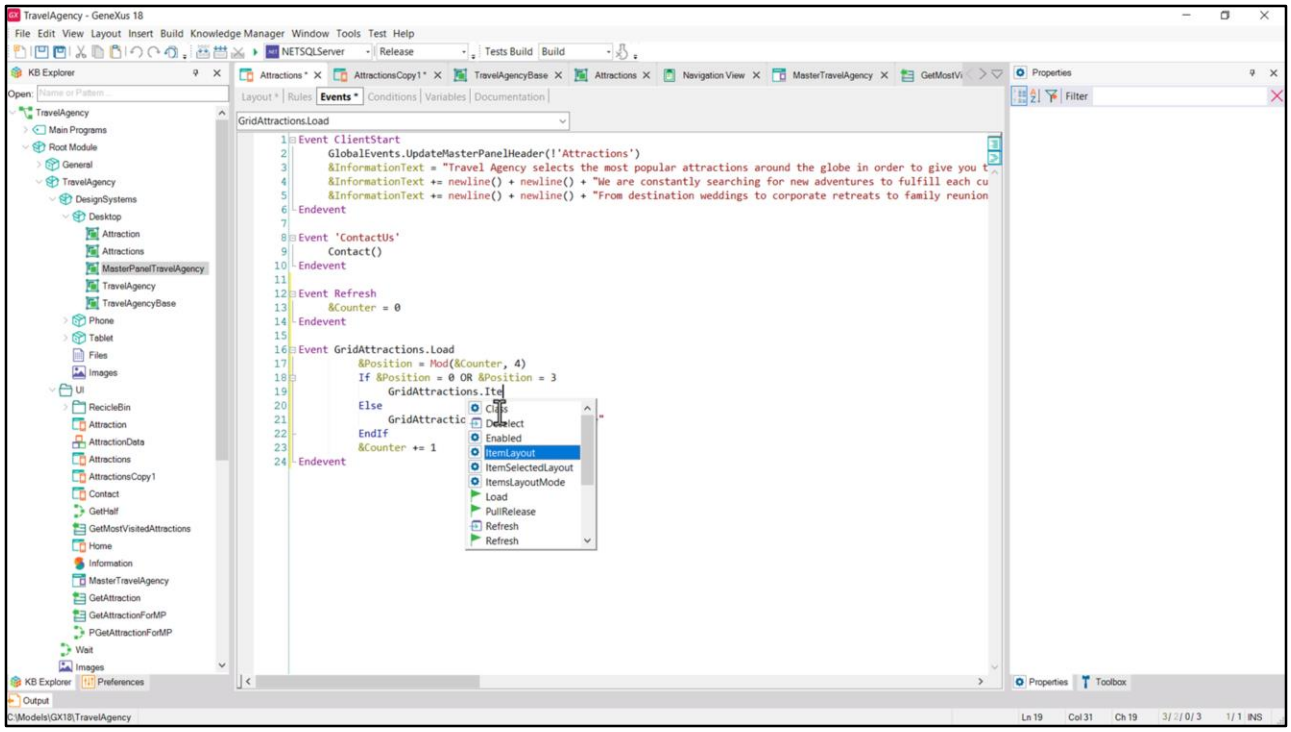
Nos resta copiar o que fizemos aqui para carregar cada item no grid com o layout que lhe corresponda.

Copio os dois eventos…

Eu colo eles… removo isso que não vai aqui. Deixo no Refresh apenas o colocar em 0 o contador…

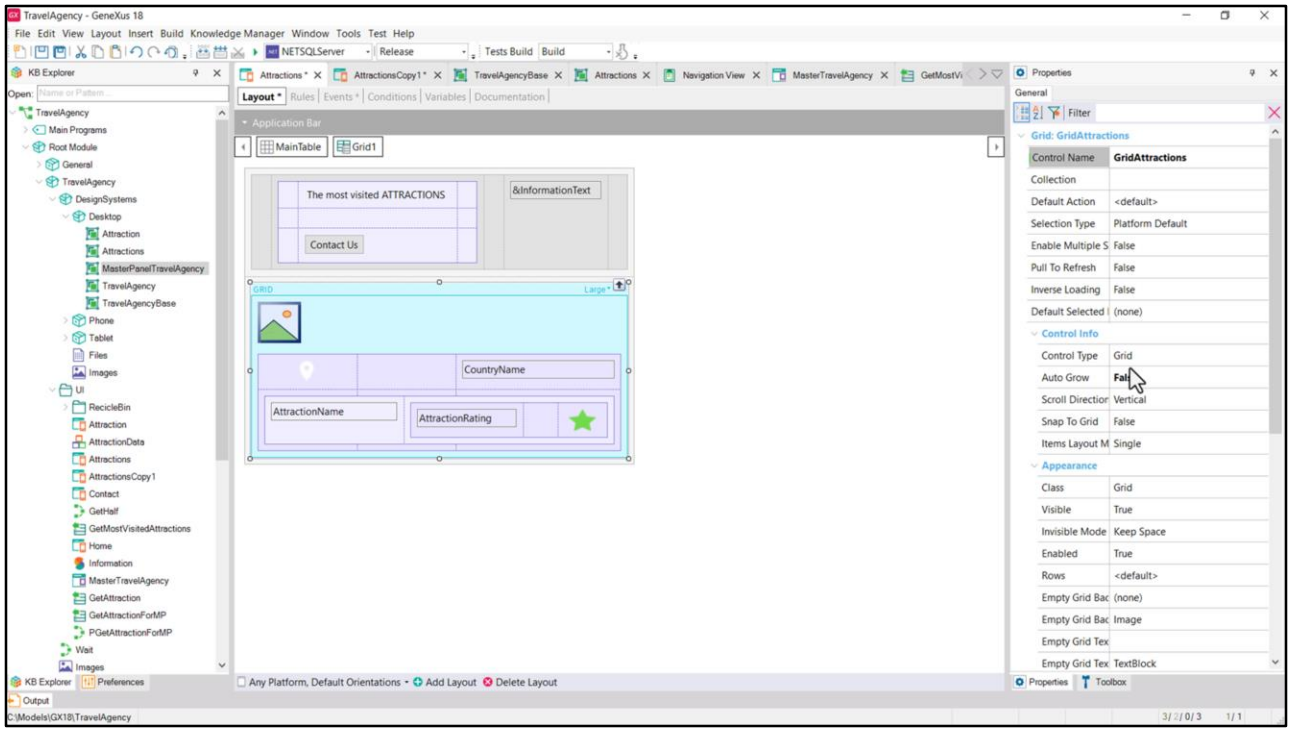


Mudo o nome do Grid para o que estava usando… GridAttractions.

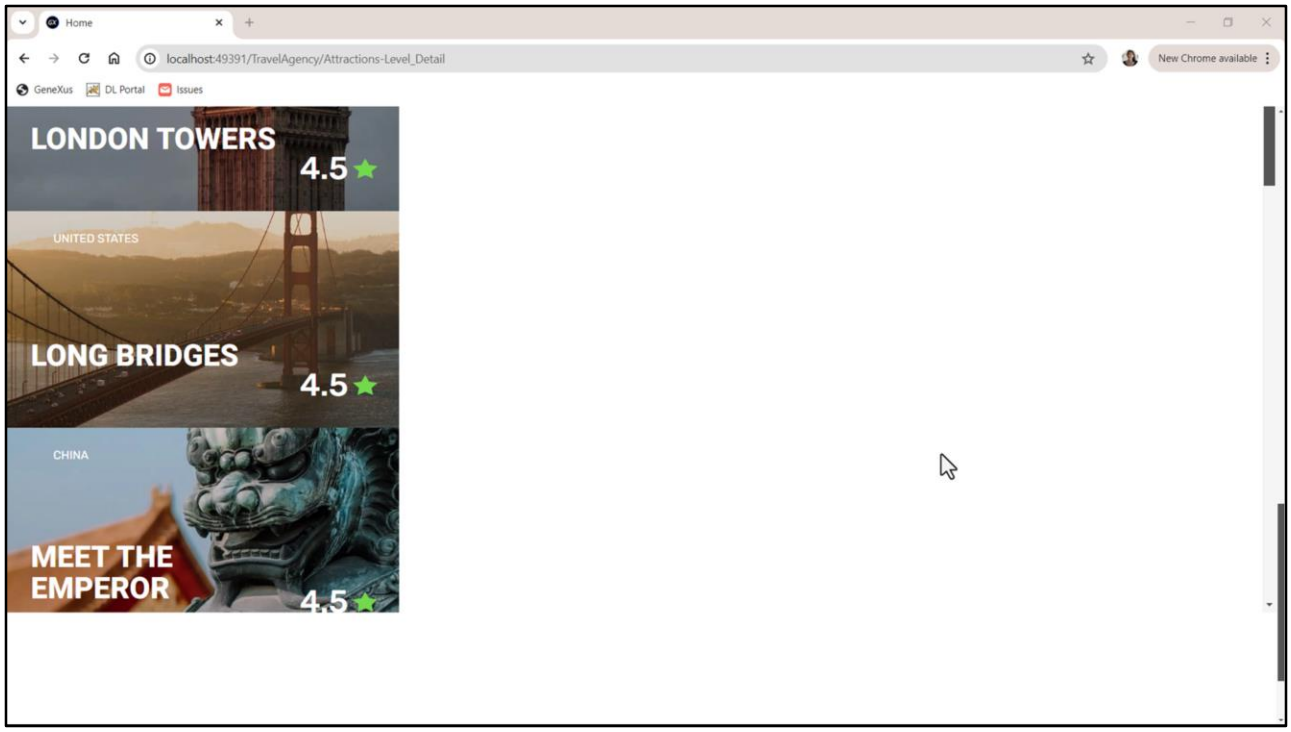


Defino neste panel as variáveis.

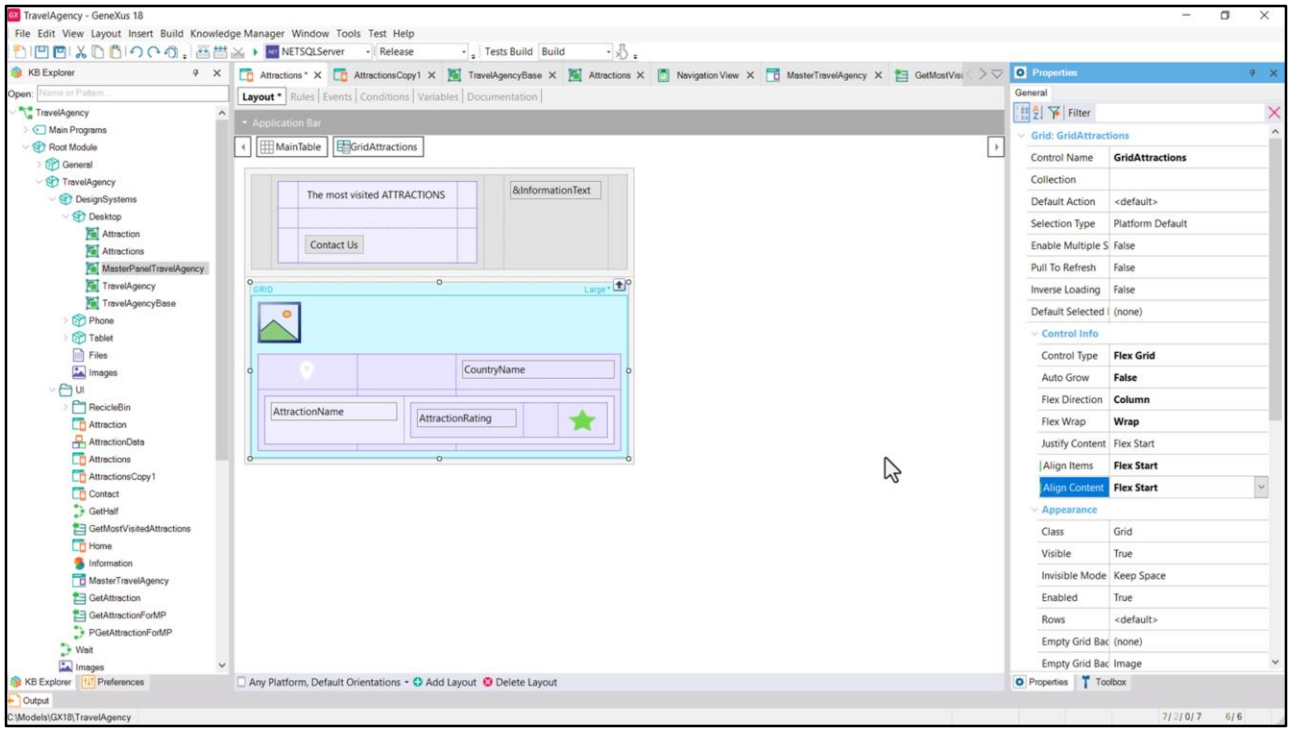
E isso fica assim, perfeito, mas para mostrar a vocês… vejam que se eu colocar um ponto depois do nome do grid… encontro elegível a propriedade… E ali eu completo. Vamos colocar o ponto de exclamação para que não sejam traduzidos este texto e o outro.



Antes de executar, vejamos que temos o grid como grid padrão. Se executo…

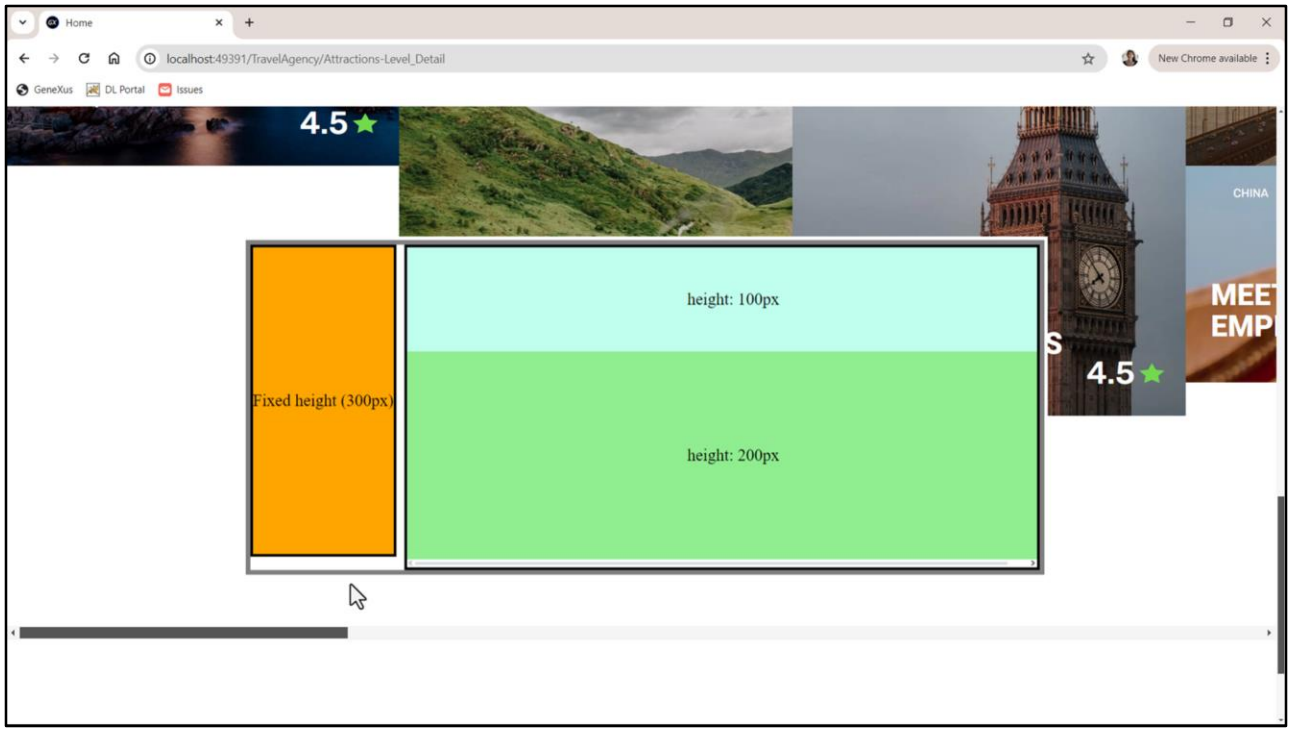


…isso é o que vejo.



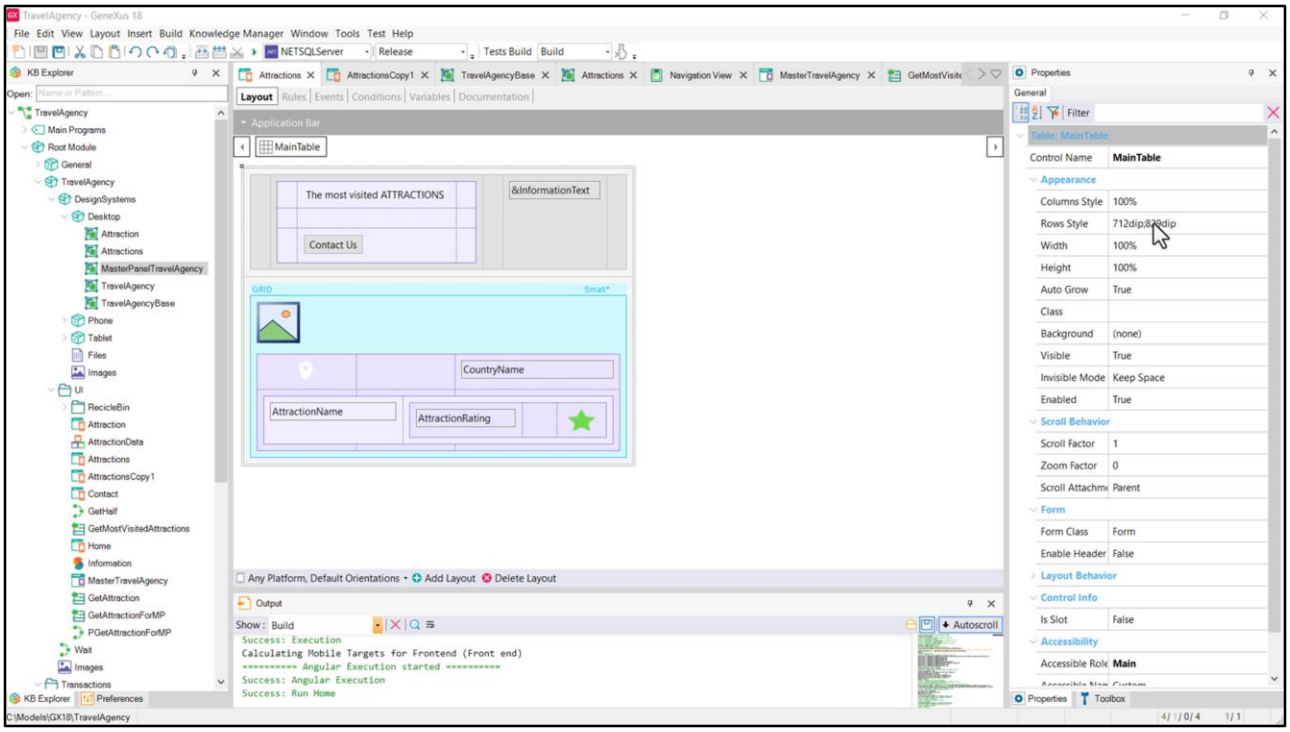
Vou modificar o grid para que seja Flex. Coloco para ele direção Column, Wrap, a justificação do conteúdo de acordo com o início, ou seja, para a borda de cima; o alinhamento dos itens em relação ao outro eixo, o horizontal, também que seja de acordo com o início, ou seja, para a borda da esquerda… E este também.

Executamos…

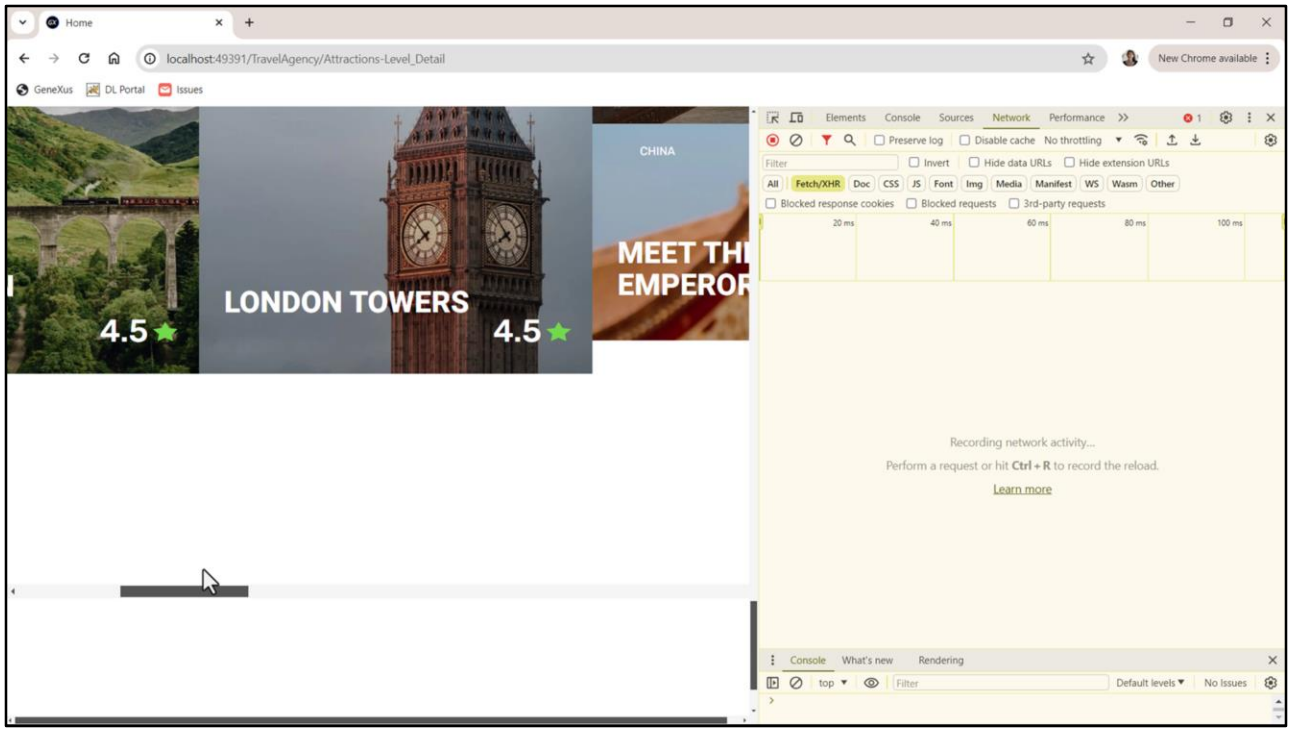


Vemos que em cada coluna está ficando apenas um card em vez de 2. Por quê?
Tem a ver com essa barra de scroll, que ocupa espaço separado em Desktop e tem que levar em conta esse espaço.

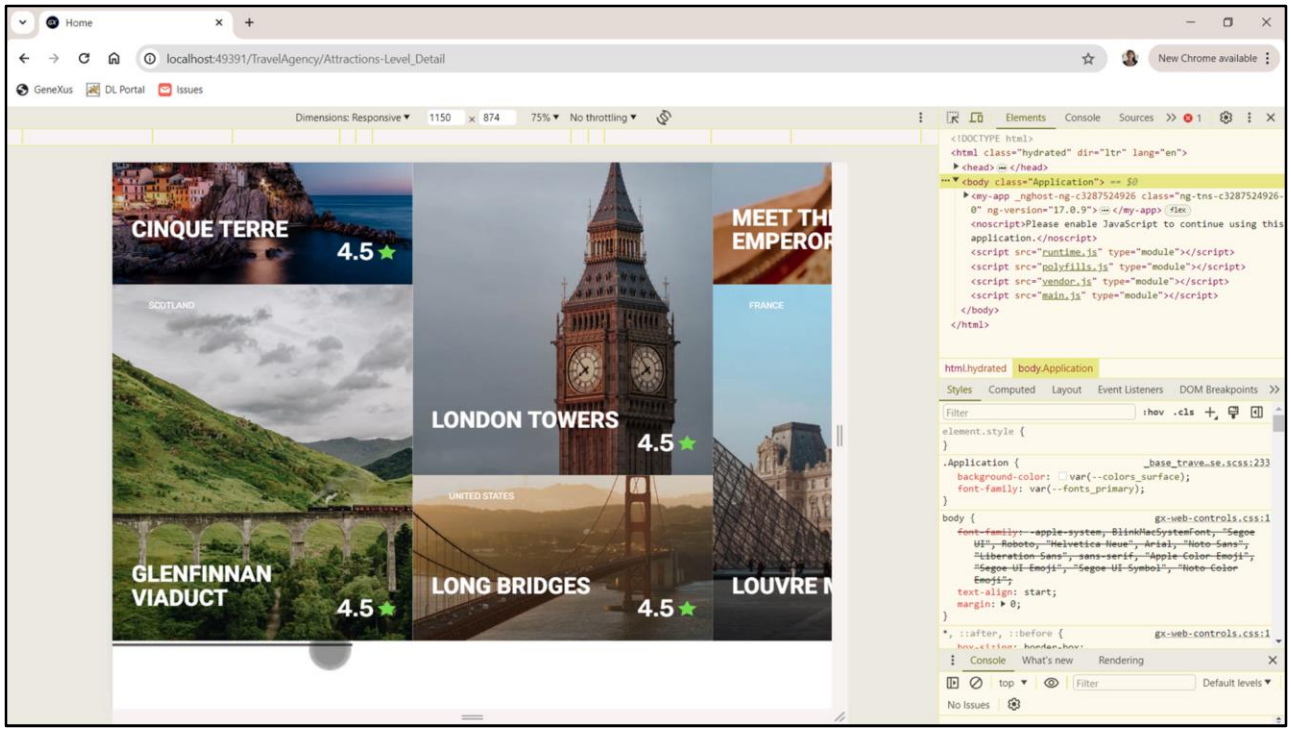
Então, por exemplo, se no grid temos um item de 100 pixels de altura e outro de 200 pixels de altura, para ambos caberem, a altura do grid não pode ser de 300, tem que ser maior, porque tem que incluir a altura da barra de scroll, que fica por fora.



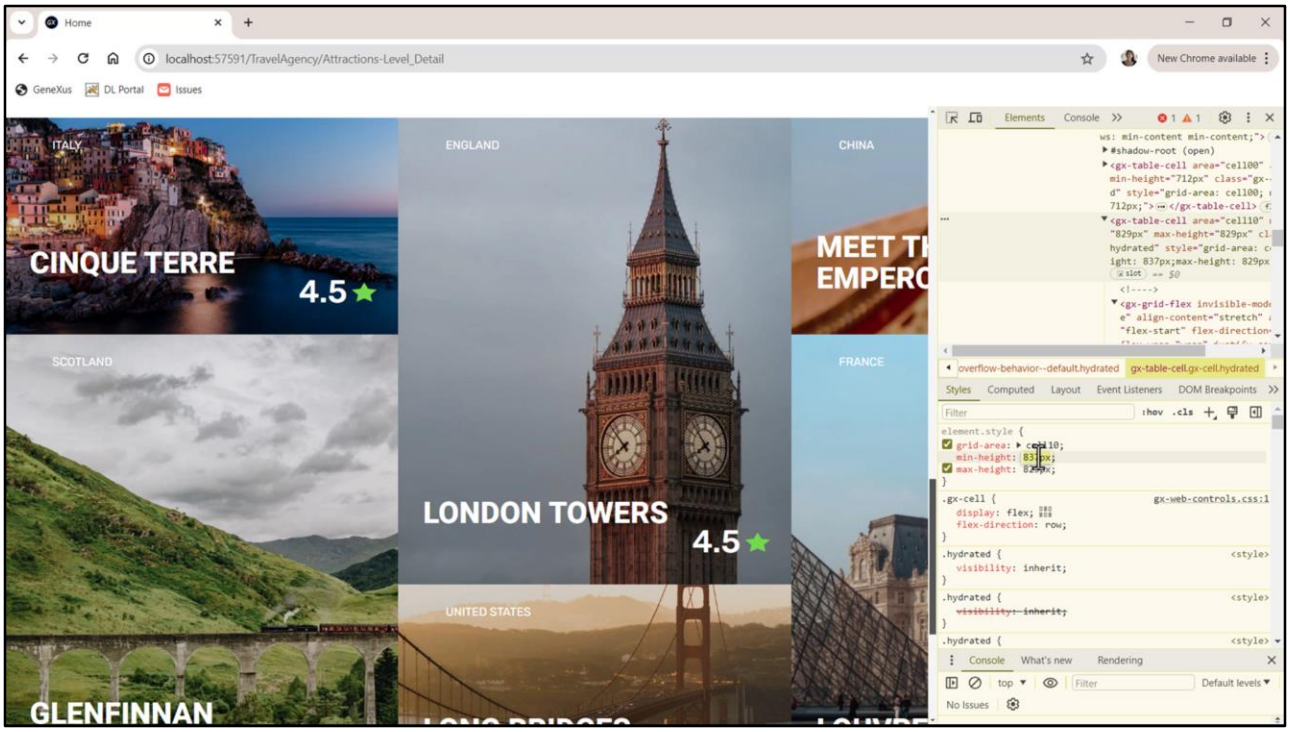
No nosso caso, a altura que demos para a linha do grid é de 829 dips, e os 2 cards somados dão 820 (260 mais 560). Obviamente, não está dando, com esses 9 dips que sobram, para colocar a barra de scroll, e por isso não estão cabendo os dois cards por coluna.



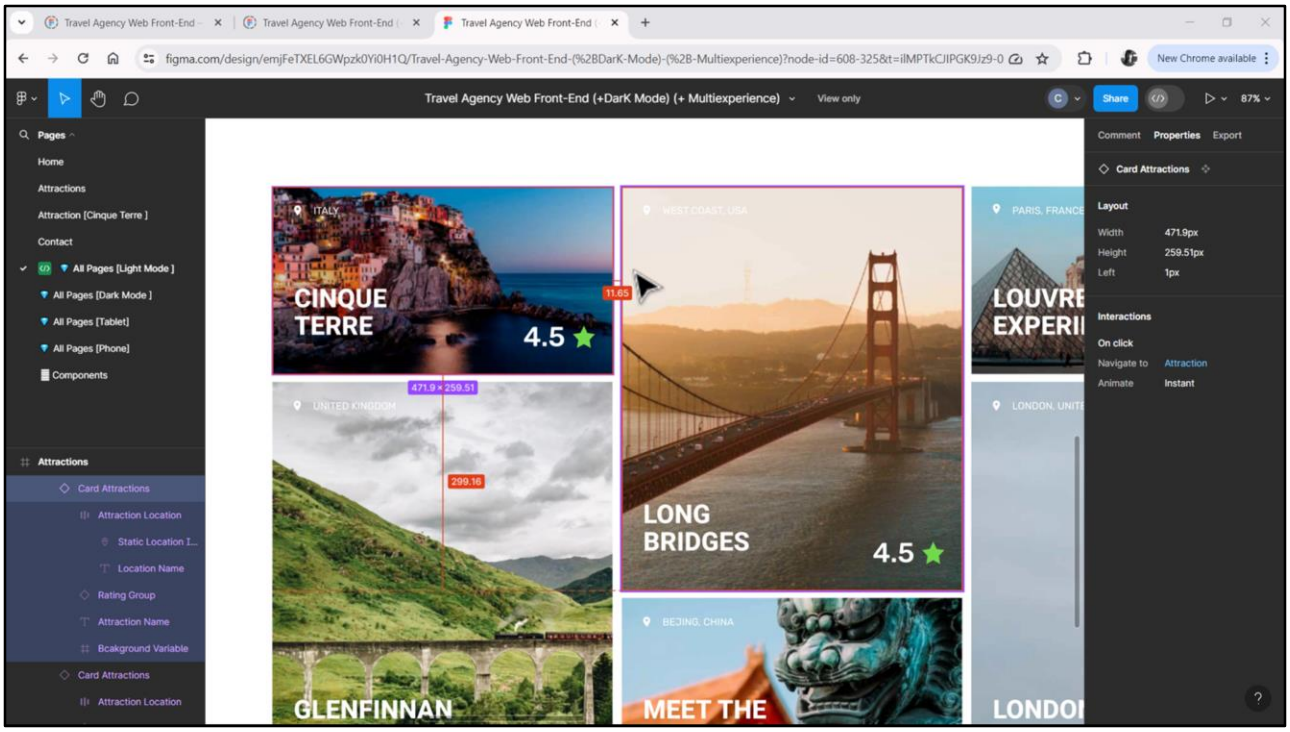
Neste F12 que lancei, só tenho habilitada a depuração Web para desktop e não a Web para dispositivo móvel.



Vejam que se ativo a depuração mobile, agora sim aparecem os dois cards, porque no navegador para mobile a barra de scroll é colocada em cima.

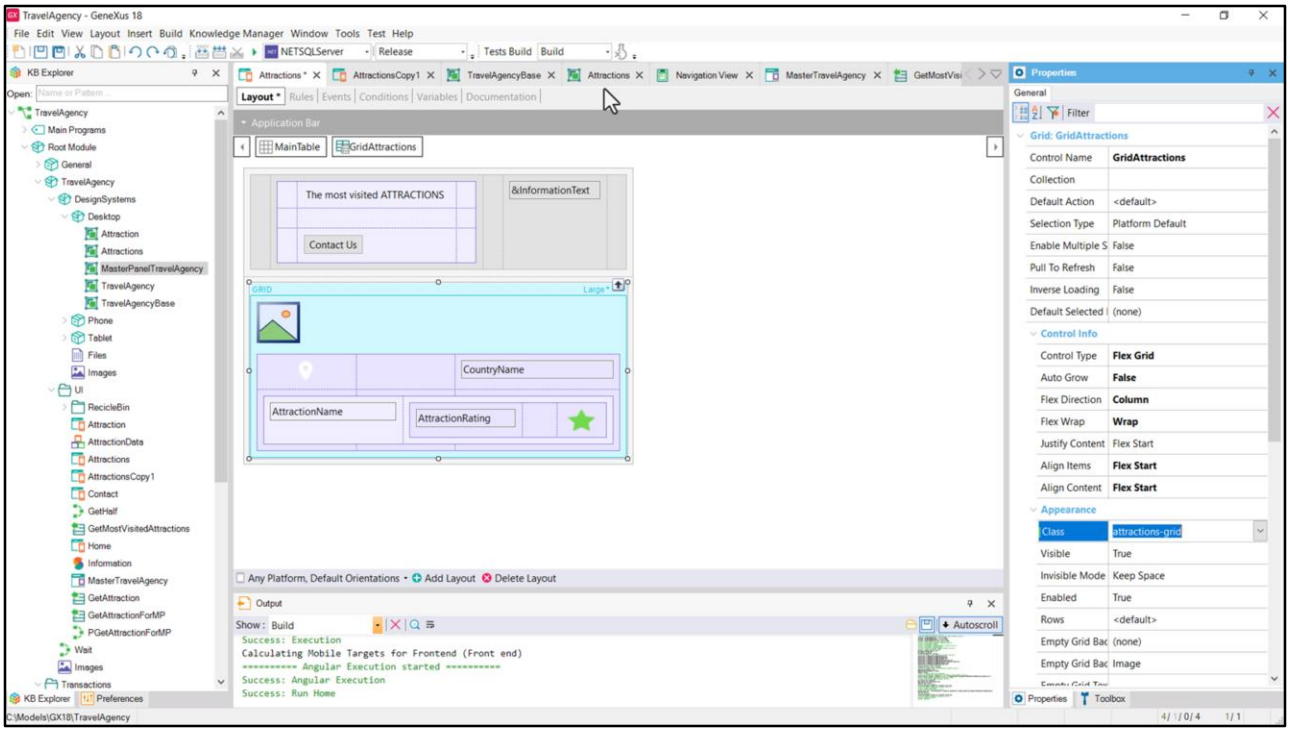


No entanto, as telas que estamos desenhando serão executadas apenas em Desktop, então precisamos resolver esse cenário. Claramente precisamos aumentar a altura da linha. Vemos que quando chegamos a esse valor, 837, aparecem os dois cards por coluna. E é claro que se continuarmos aumentando, veremos como vai se separando a barra de scroll, então precisamos dedicar, destinar para ela, 17 dips, porque os dois cards somavam 820... precisamos de 17 dips para essa barra de scroll.

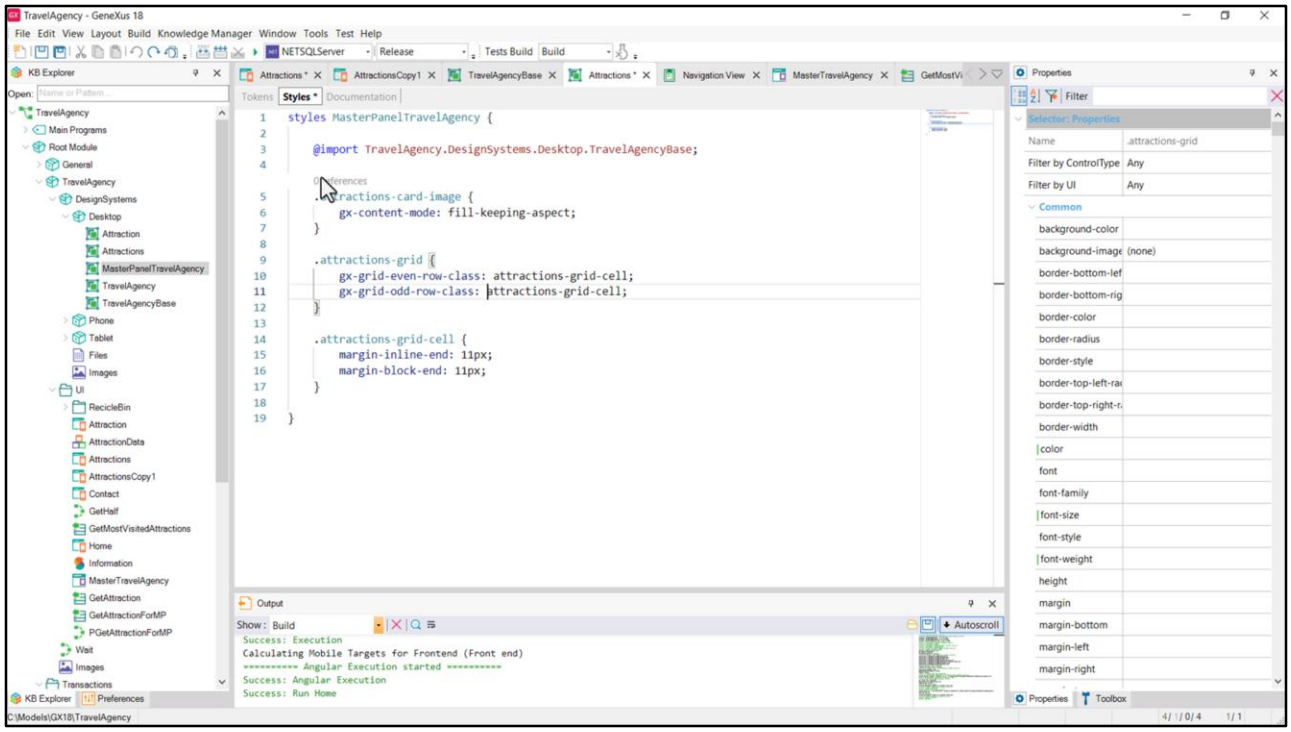


Mas também precisamos deixar o espaço vazio entre cards…

Vemos que no design de Figma os cards têm uma margem à direita de um pouco mais de 11 pixels e na parte inferior de quase 11. Então poderíamos para todos os cards no grid, atribuir uma margem de 11 final, nas duas direções.



Podemos atribuir uma classe ao grid…

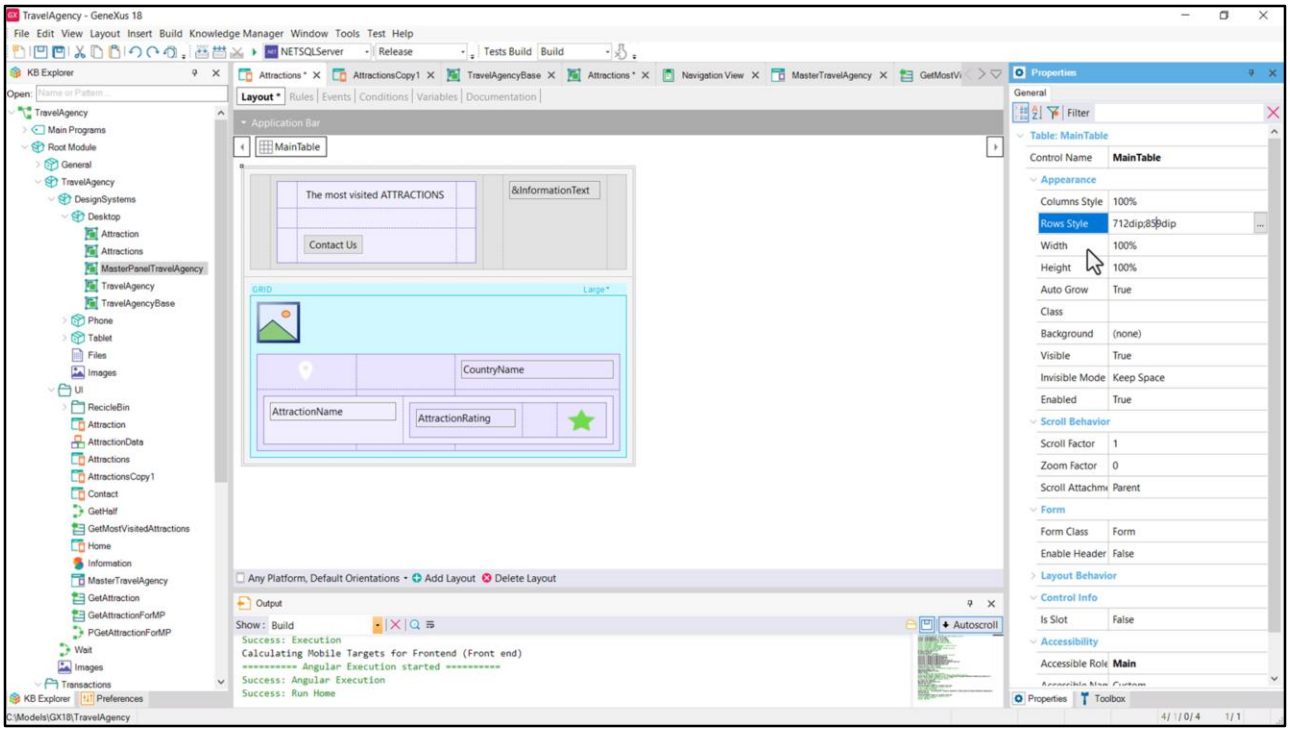


…na qual definimos através das gx-properties: gx-grid-**even**-row-class e gx-grid-**odd**-row-class, classes que dão estilo à célula do grid quando é um item par ou item ímpar.

Esta row aqui no nome, linha, deve ser entendida como posição do item. Ou seja, além de como são apresentados os itens renderizados na tela, algo que depende, entre outras coisas, do tipo de grid, todo grid é conceitualmente uma lista ordenada de itens. Então vem o primeiro, o segundo, o terceiro e assim por diante. Depois tem que ver como essa ordem é renderizada, como dizia. A propriedade gx-grid-**even**-row-class será aplicada a todos os itens que ocupam posição **par** nessa lista, e a **odd** para aqueles que ocupam posição **ímpar**. Nos permite essa discriminação, digamos, essa alternância entre pares e ímpares.

No nosso caso, queremos que a todos os itens se aplique a mesma classe, que chamarei de attractions-grid-cell e definirei essas duas propriedades: margin-inline-end de 11 pixels e margin-block-end, ou seja, na outra direção, também de 11 pixels.

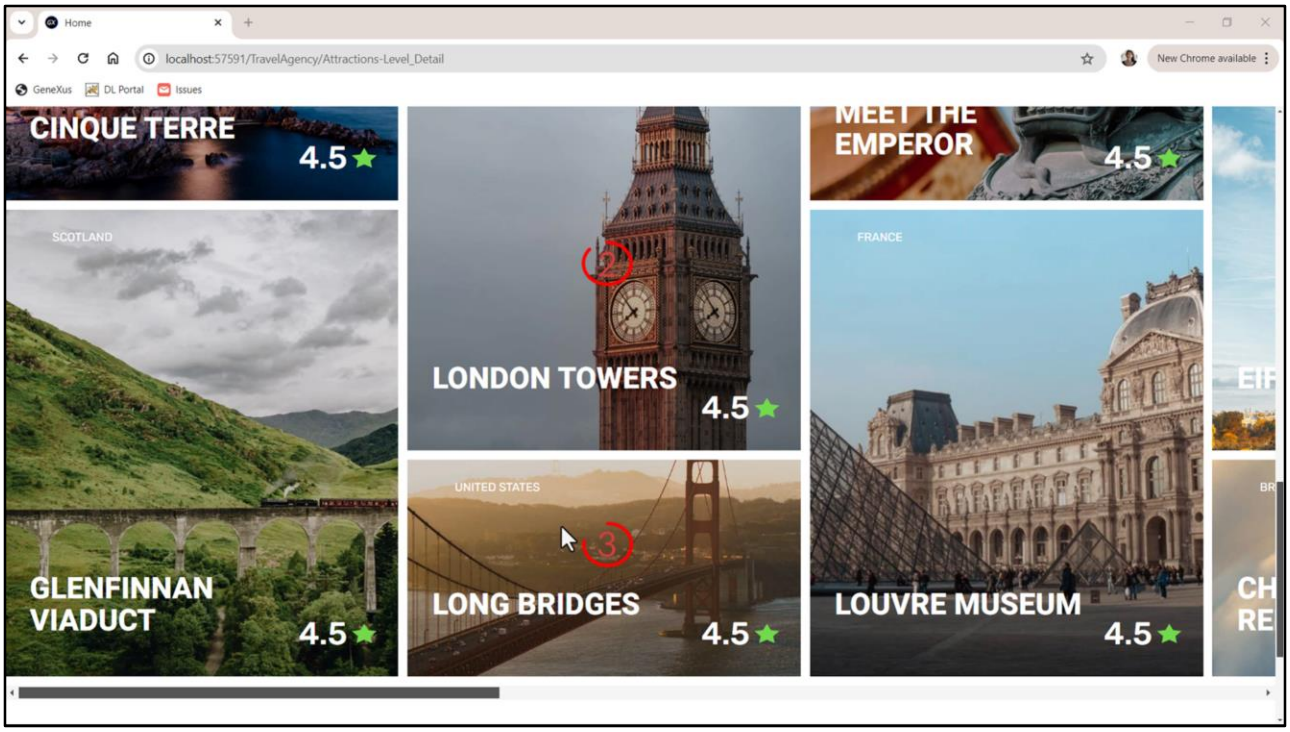
E então simplesmente estabeleço que a classe para os itens pares e para os ímpares é a mesma, e é esta.



O que preciso fazer agora é calcular a altura que devo dar para a linha do grid. Será de 11 x 2, devido à margem block end dos dois cards mais os 820 de altura dos dois cards somados, mais os 17 da barra de scroll.

Então… vou em Attractions… e na segunda linha… coloco esse valor.

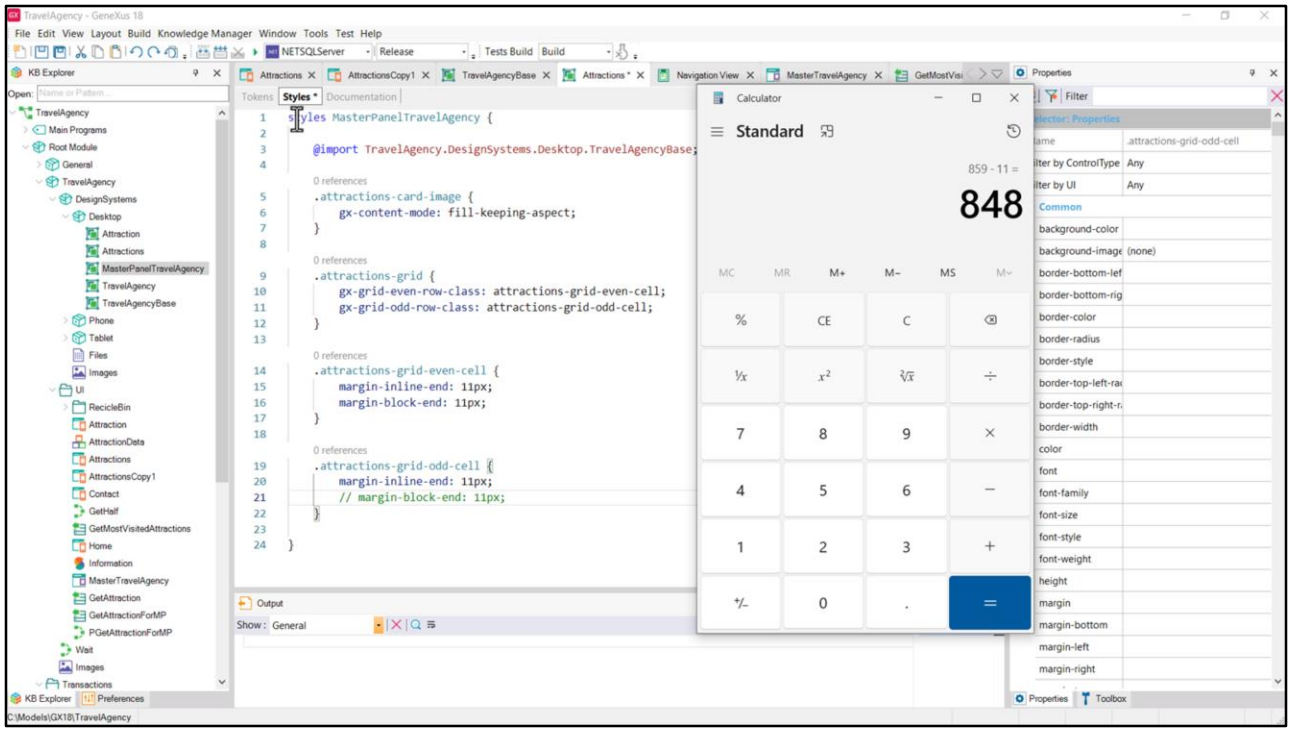
Executemos.



Perfeito.

E se não quiséssemos a margem inferior para os cards de baixo?

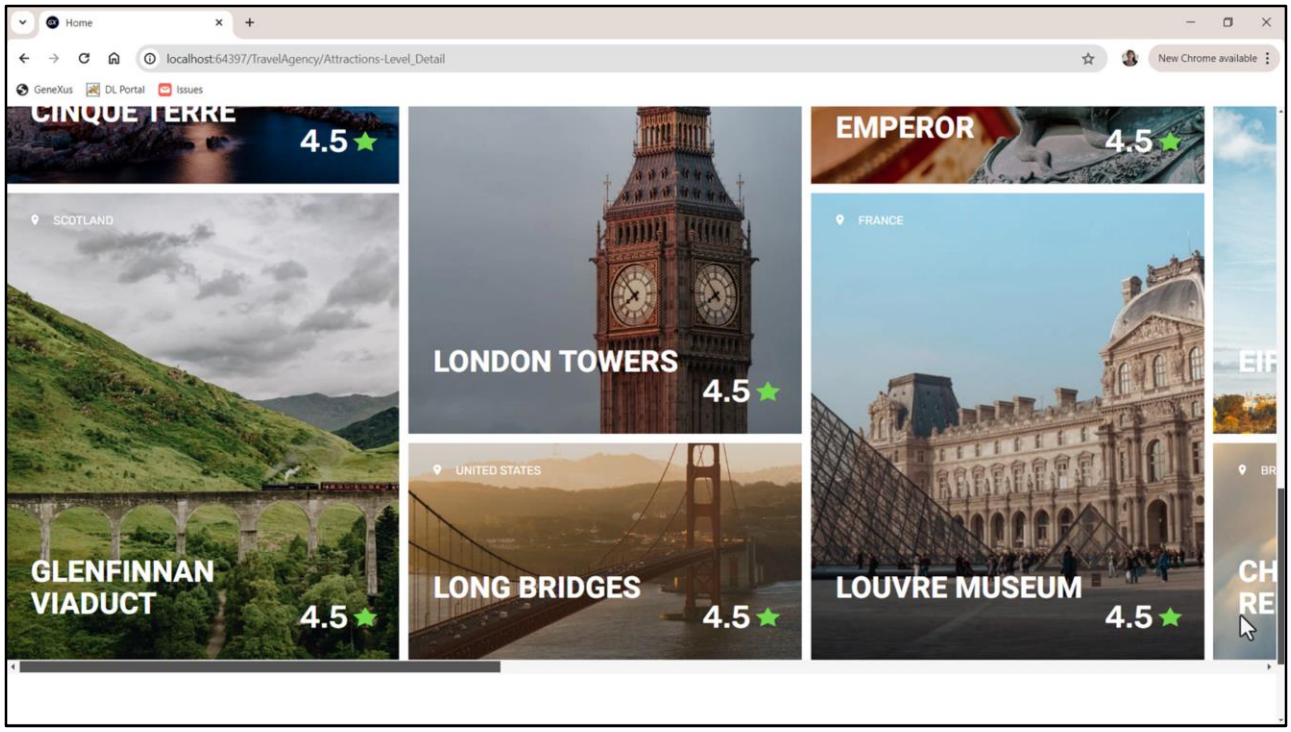
Os cards de baixo serão os ímpares, começar a contar do 0.
0, 1… 2, 3… 4, 5… e assim por diante.



Então podemos fazer isso…

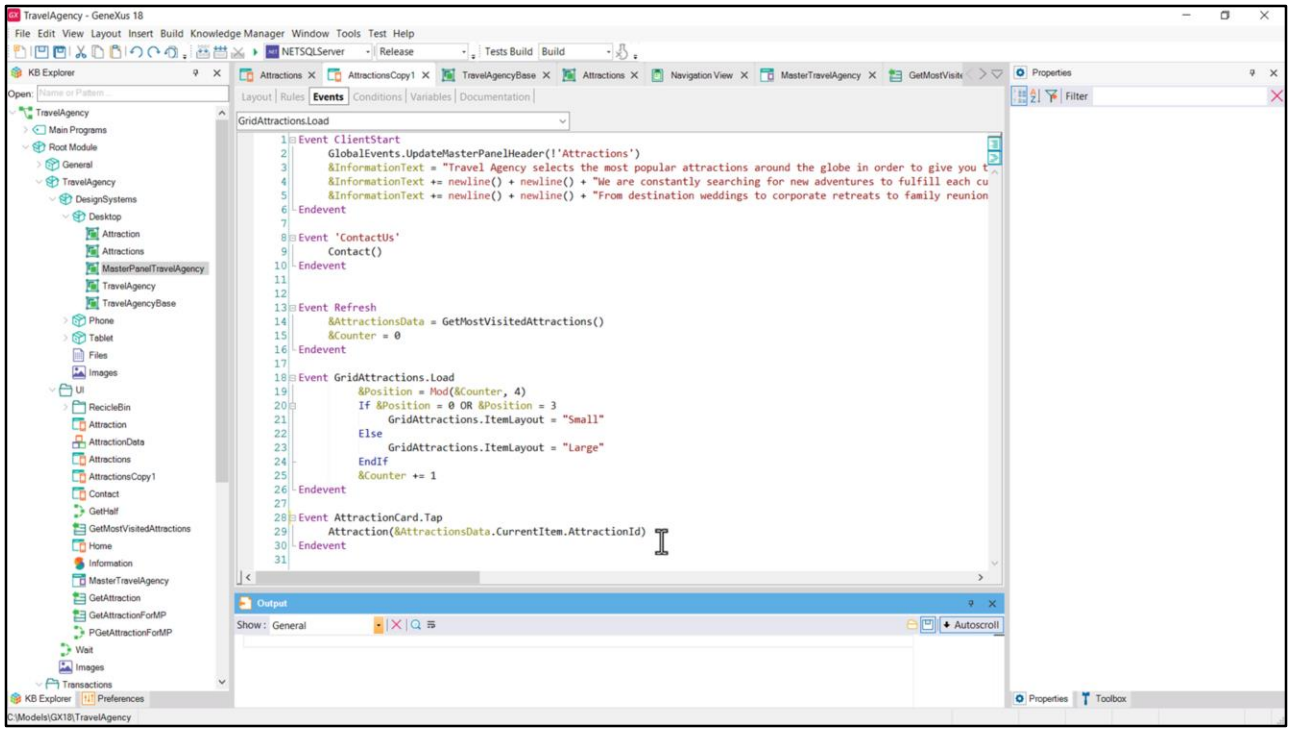
Diferenciamos duas classes: uma para os pares e outra para os ímpares, e para a dos ímpares removemos a margem inferior.

E agora subtraímos da altura da linha esses 11



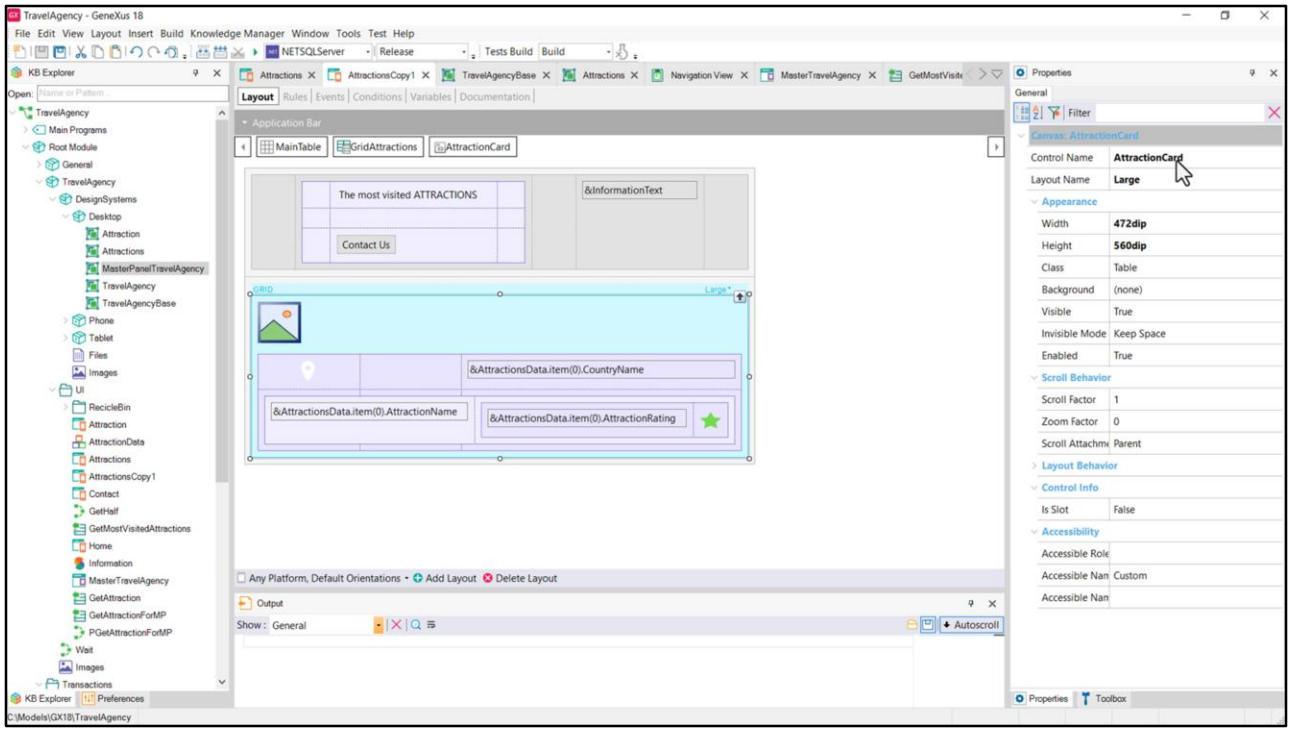
e executamos…

Bem, vemos que a margem inferior desapareceu.

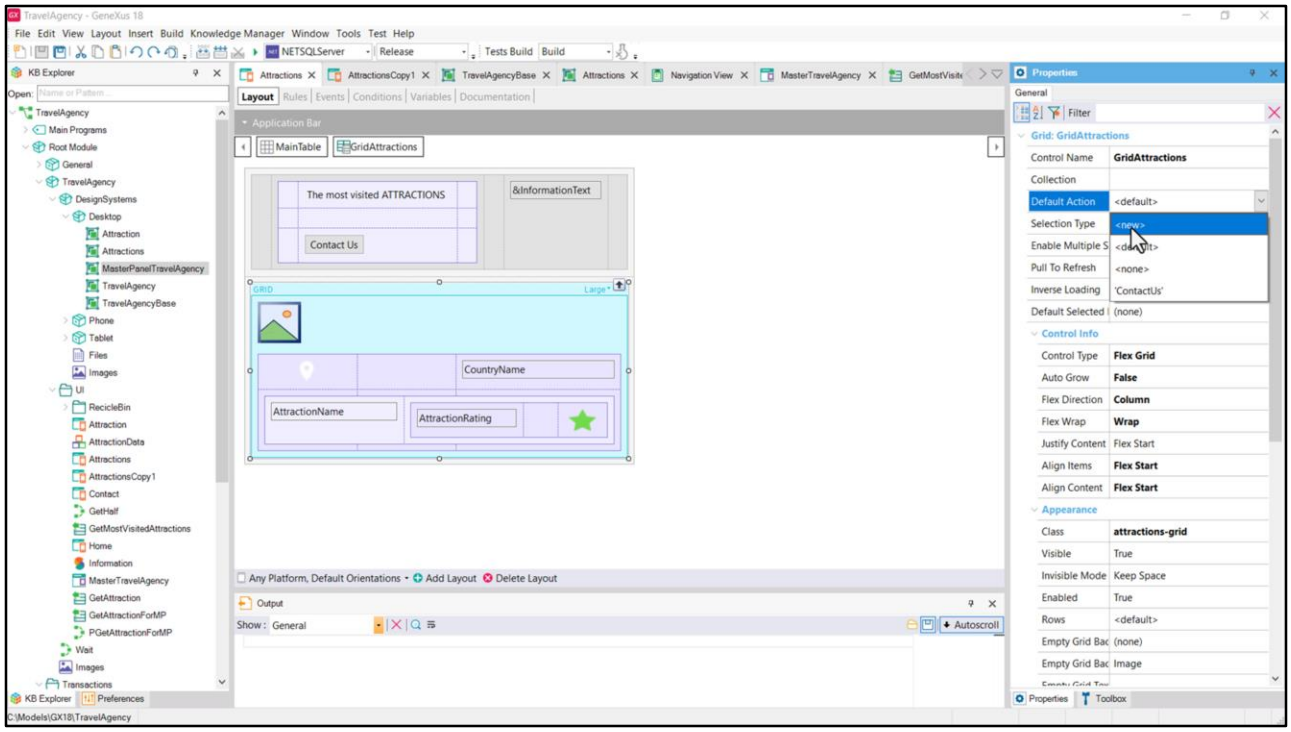


Na minha solução com o grid associado ao SDT, vamos fazer o mesmo…

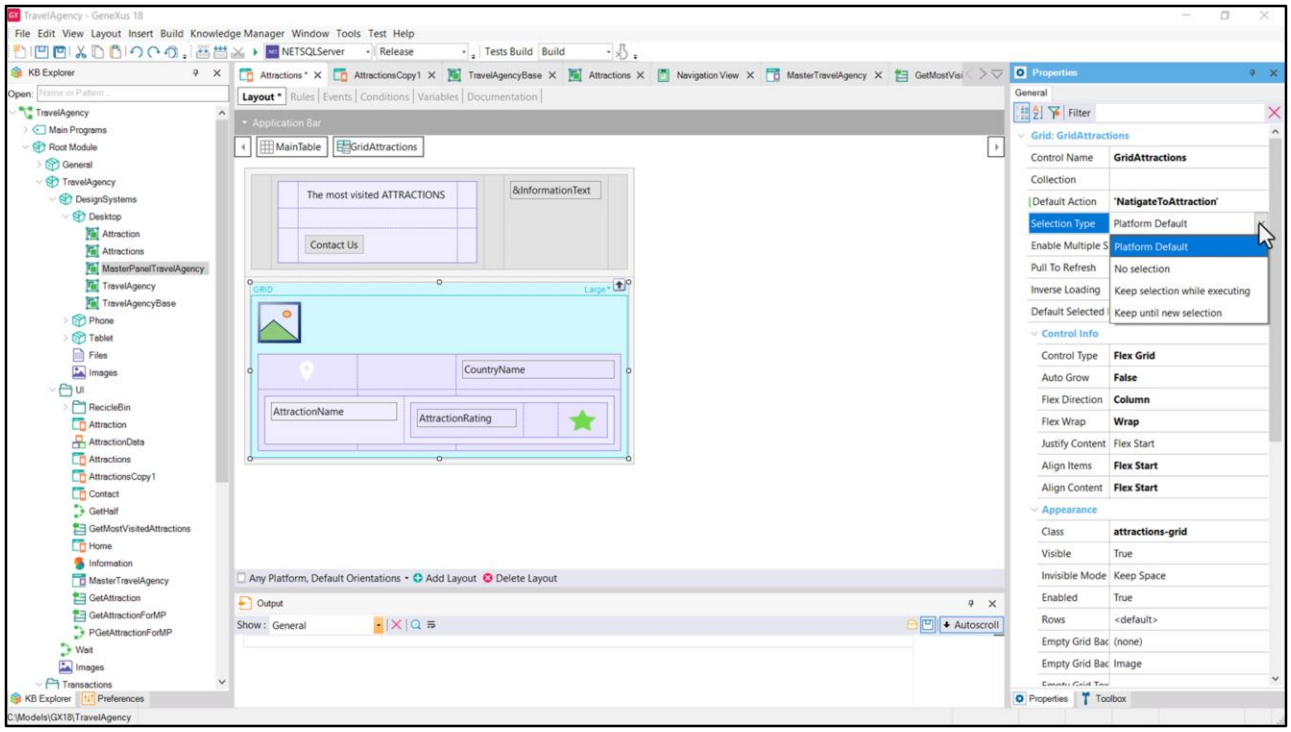
Agora vamos dar uma olhada neste evento Tap que associamos ao Canvas de cada item para poder invocar o panel Attraction passando a ele o id da atração.



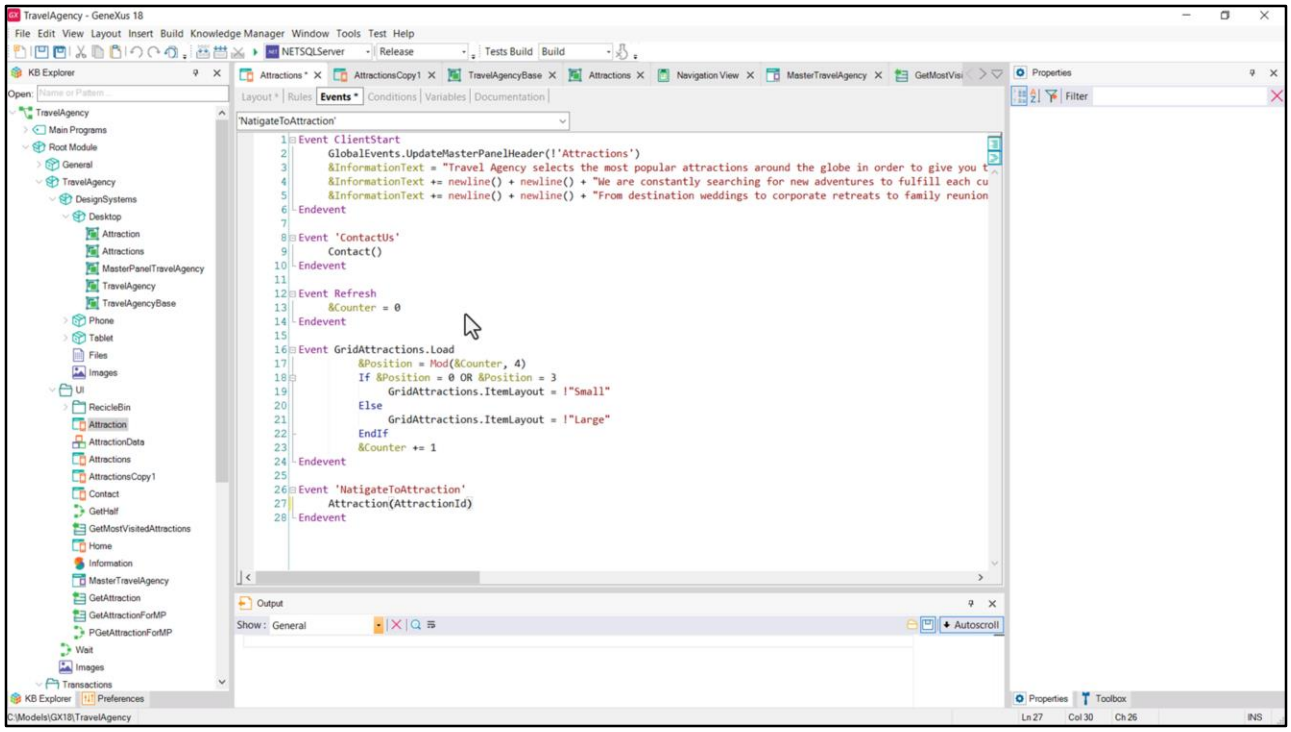
Esta não é a melhor solução, porque temos o tap programado para o controle chamado AttractionCard, que na verdade corresponde a dois canvas diferentes: o do layout Small e o do layout Large (tive, na verdade, que dar explicitamente o mesmo nome, ter esse cuidado).



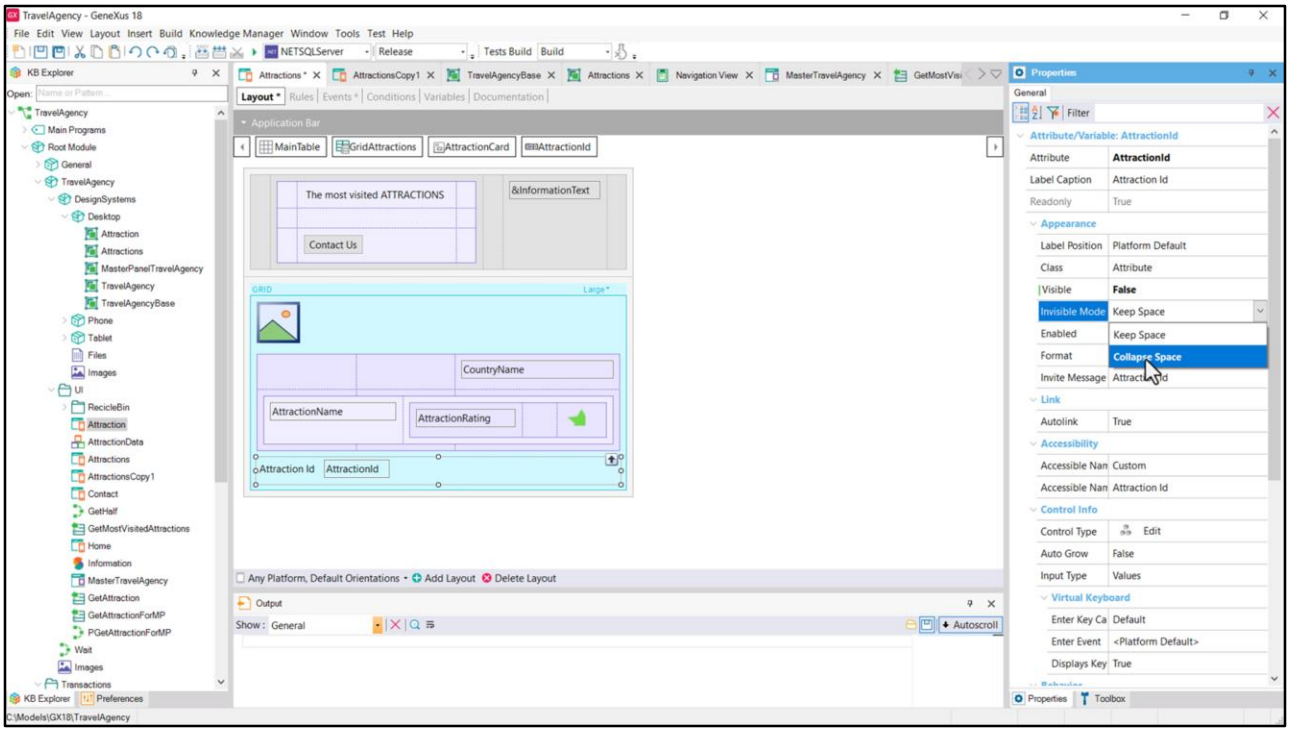
Na verdade, o grid tem um evento predefinido que corresponde a fazer tap ou click sobre qualquer um de seus itens. Corresponde a esta propriedade, Default Action. Usaremos essa outra solução em nosso panel Attraction.
O que eu faço é criar um novo evento de usuário, que corresponderá à ação default sobre os itens do layout. Dou um nome….



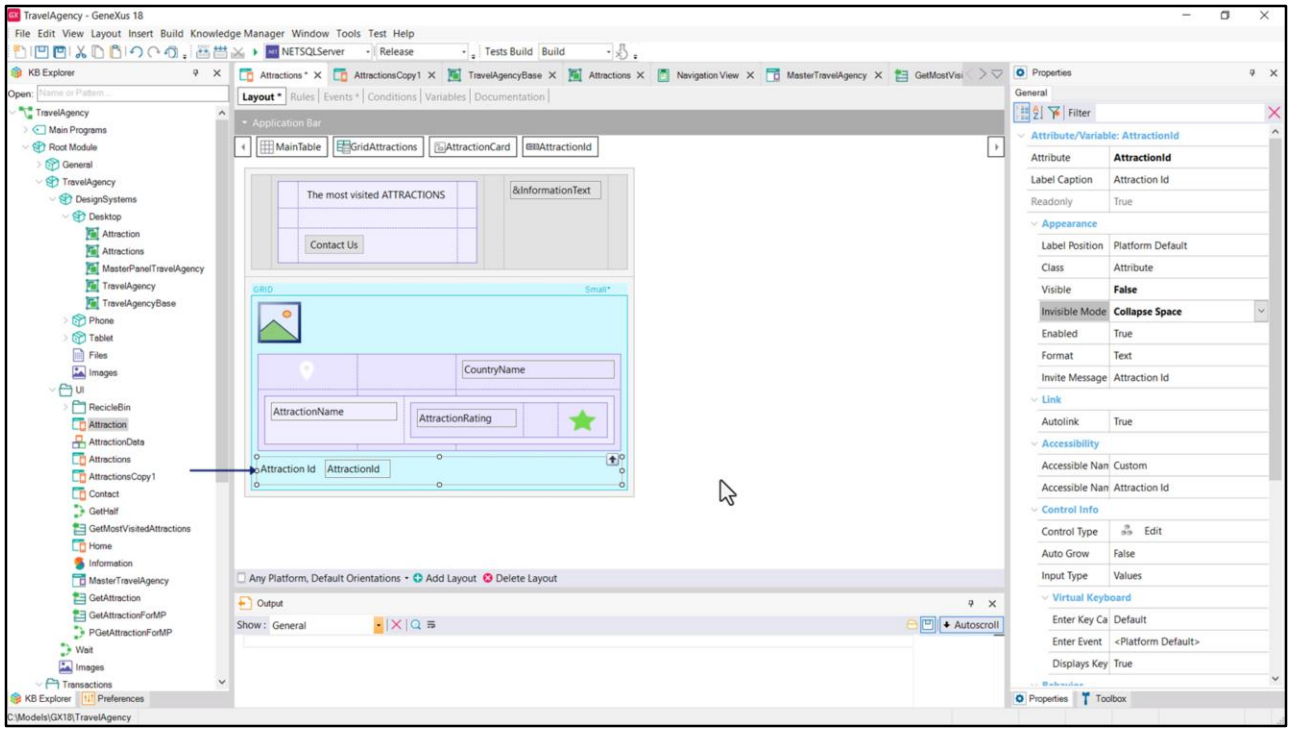
Aproveitamos para ver que há um conjunto de propriedades que permitem definir determinado comportamento do grid... por exemplo, se permite selecionar os itens do grid, e que tipo de seleção é permitida, etc.



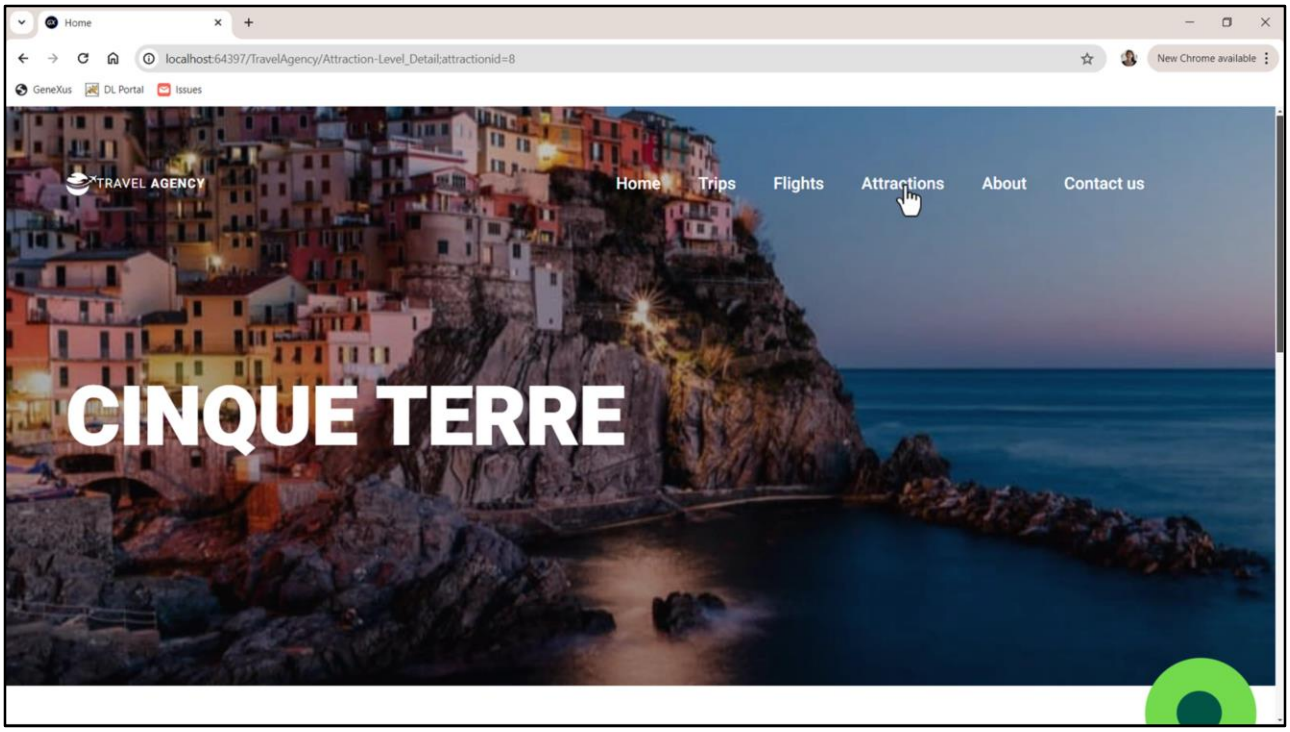
Aqui está o evento, e o que precisamos fazer é invocar o panel Attraction, passando o identificador de atração, que neste caso estará no atributo AttractionId. Mas temos carregado no item do grid o atributo para que possamos passá-lo para esse outro panel quando o grid for carregado?



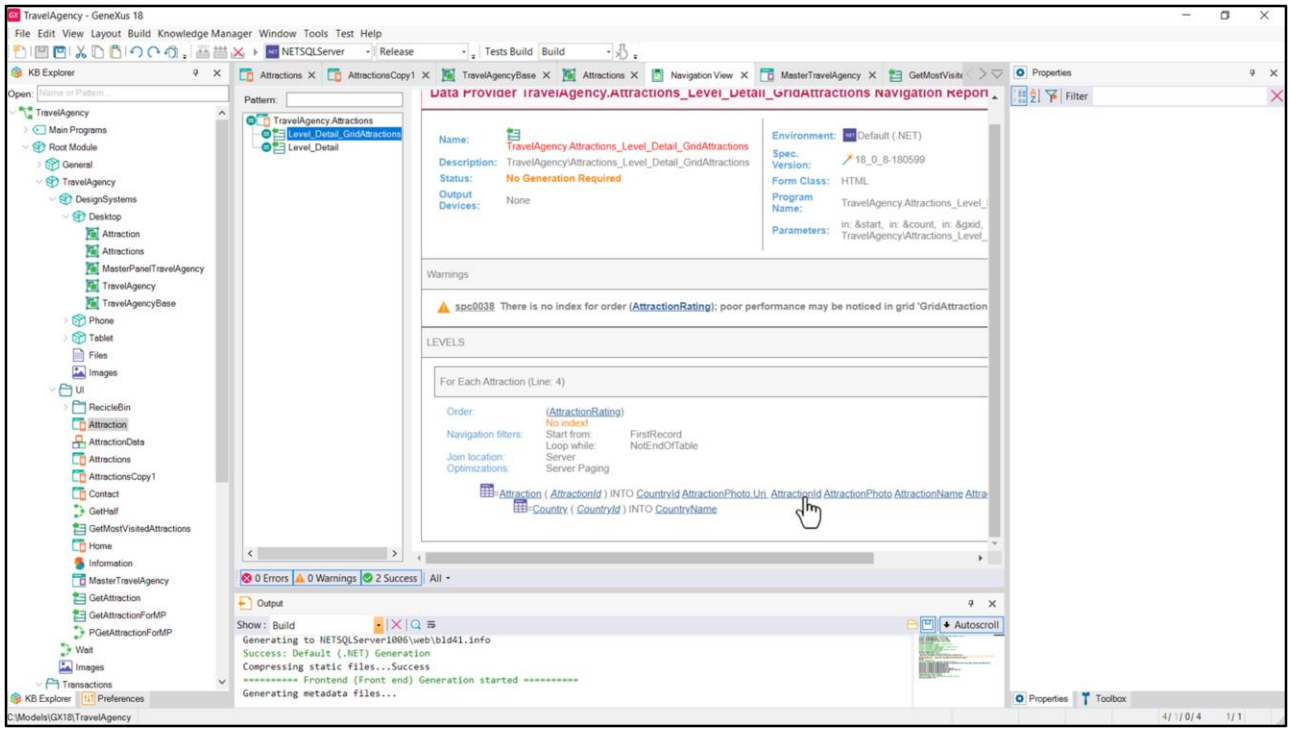
A priori parece que não. Então, se quisermos ter certeza de que se conte com esse valor, podemos inserir o atributo, torná-lo invisível e também dizer que não seja reservado espaço para ele no layout quando estiver invisível, justamente.



Bom, isso vale para o layout Large. Teremos que fazer o mesmo para o Small. E isso já pode começar a nos incomodar, essas duplicações, e já podemos começar a ficar tentados a criar um stencil para esses cards.
Se não fizermos isso, então terei que copiar esse controle e colá-lo aqui.

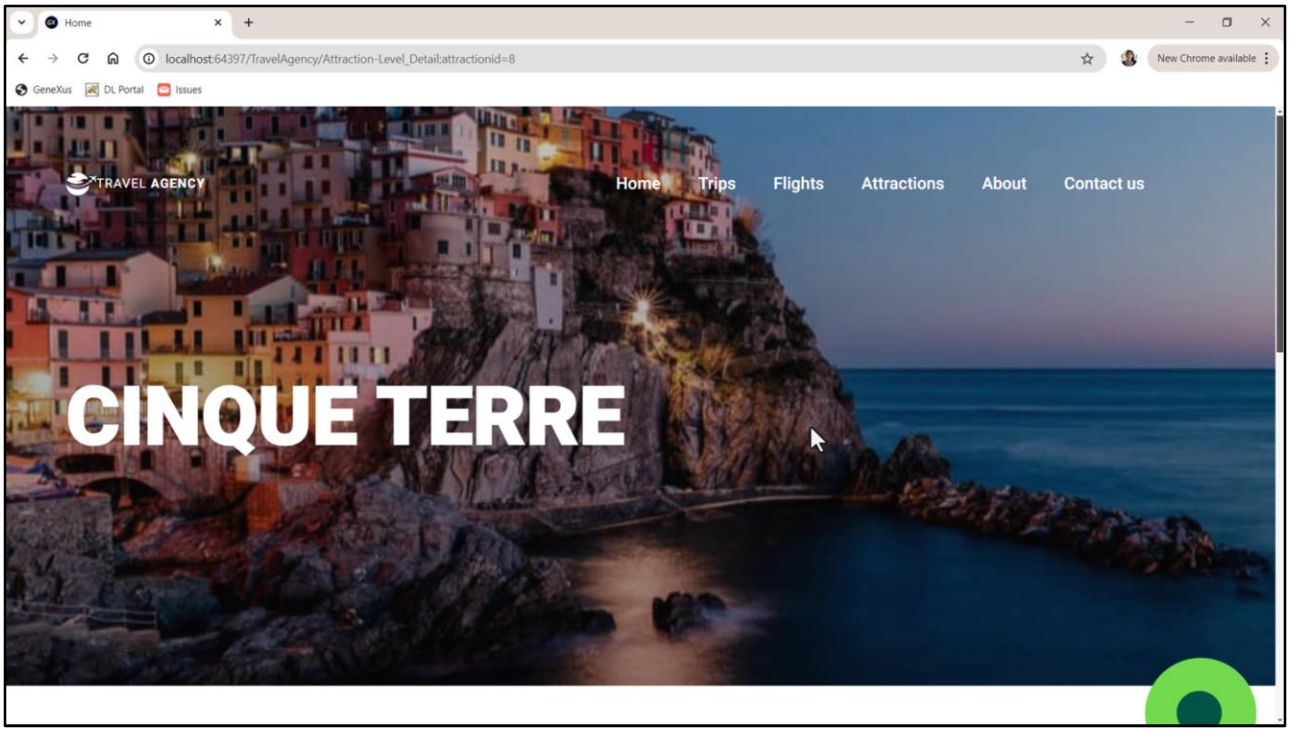


Se agora executarmos… bem… Vamos testar com outro… o large… perfeito.



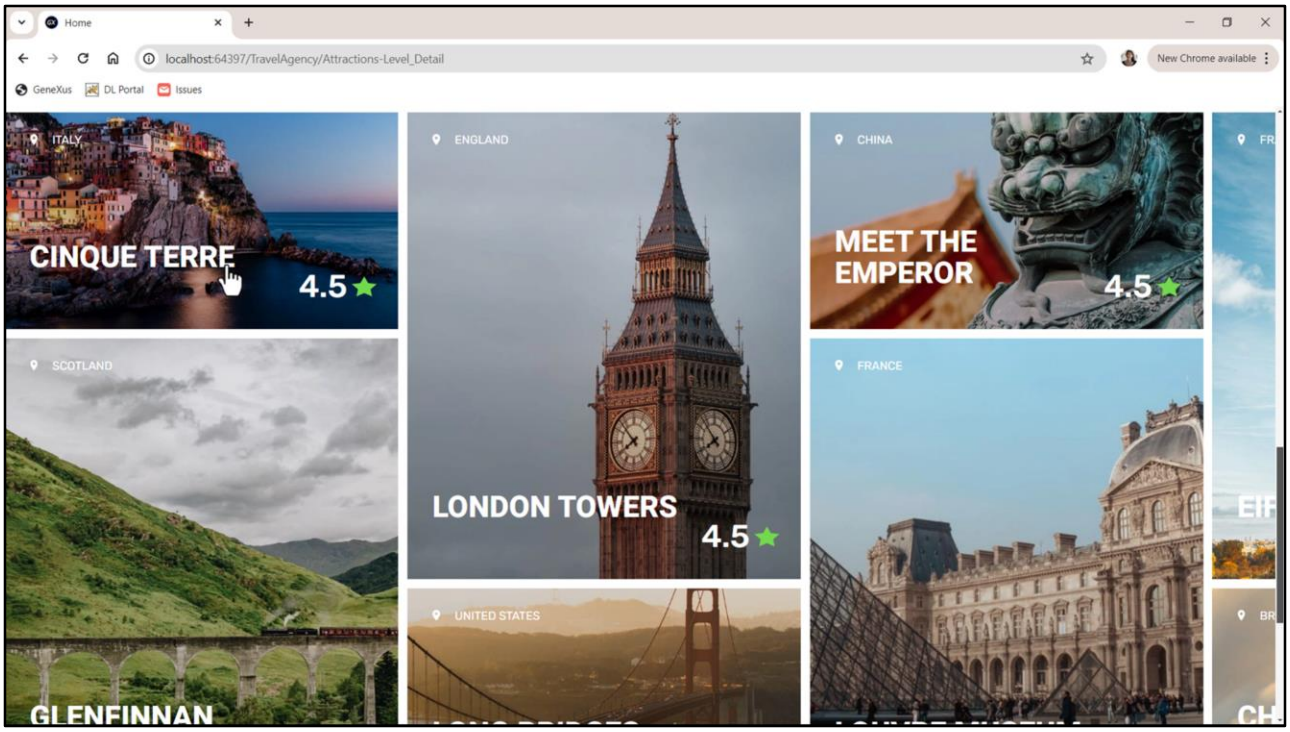
Agora vamos ver se era necessário colocar o atributo invisível. Vamos removê-lo… e removê-lo também do layout Large.

Vamos executar. Mas antes de terminar, vejamos a lista de navegação. Vemos no Data Provider que corresponde ao grid que nos informa que está trazendo AttractionId. Será armazenado internamente. Não precisávamos fazer o que fizemos.

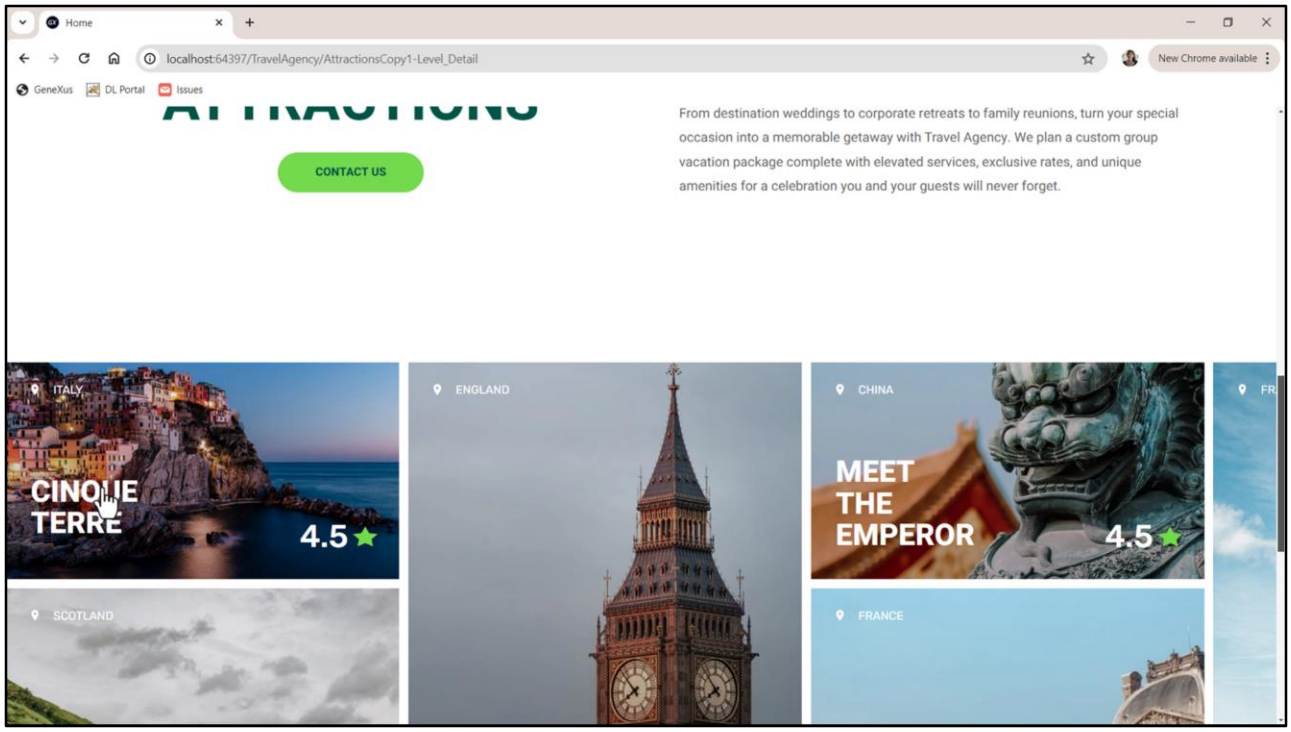


E vamos terminar de verificar isso... bem.

Está passando, efetivamente, o id de atração porque o tem e GeneXus fez isso automaticamente. Digo isso porque nos Web Panels isto não é assim.

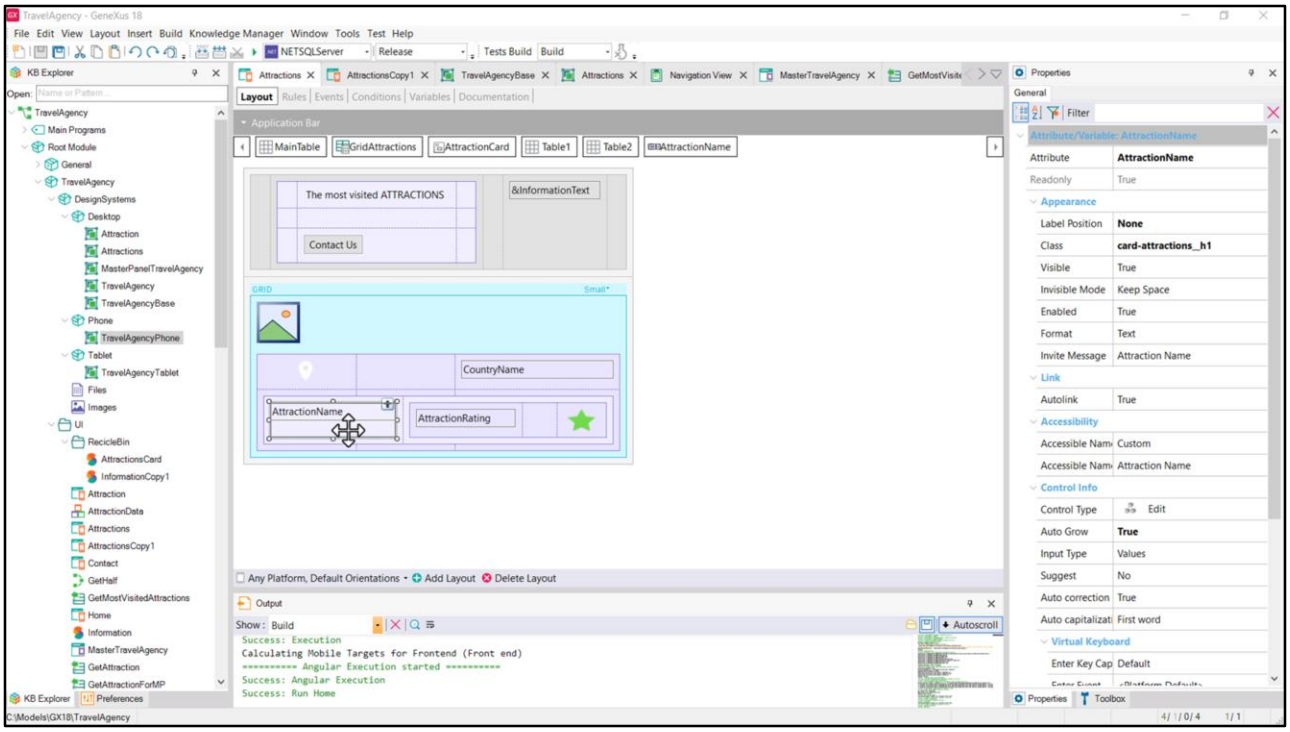


Algo que pode chamar nossa atenção é que aqui o texto não está ficando dividido em linhas…

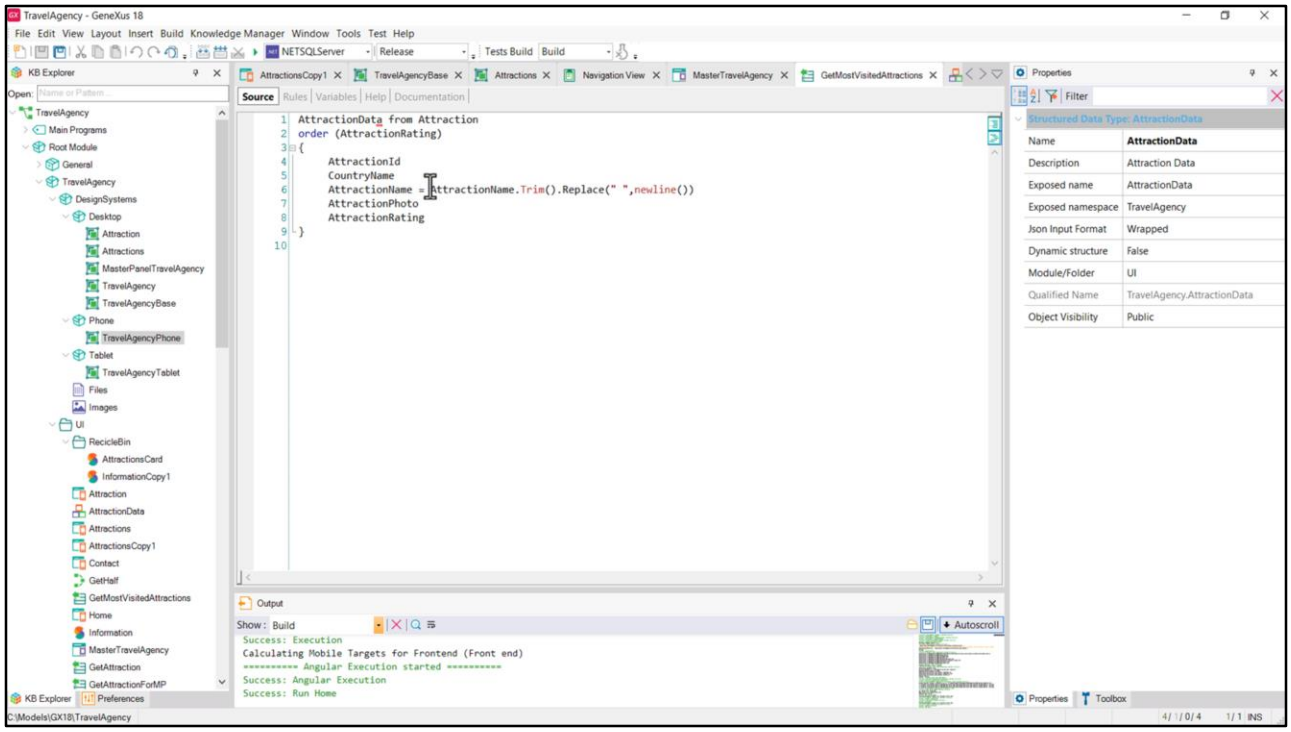


...como estava na solução com o SDT.

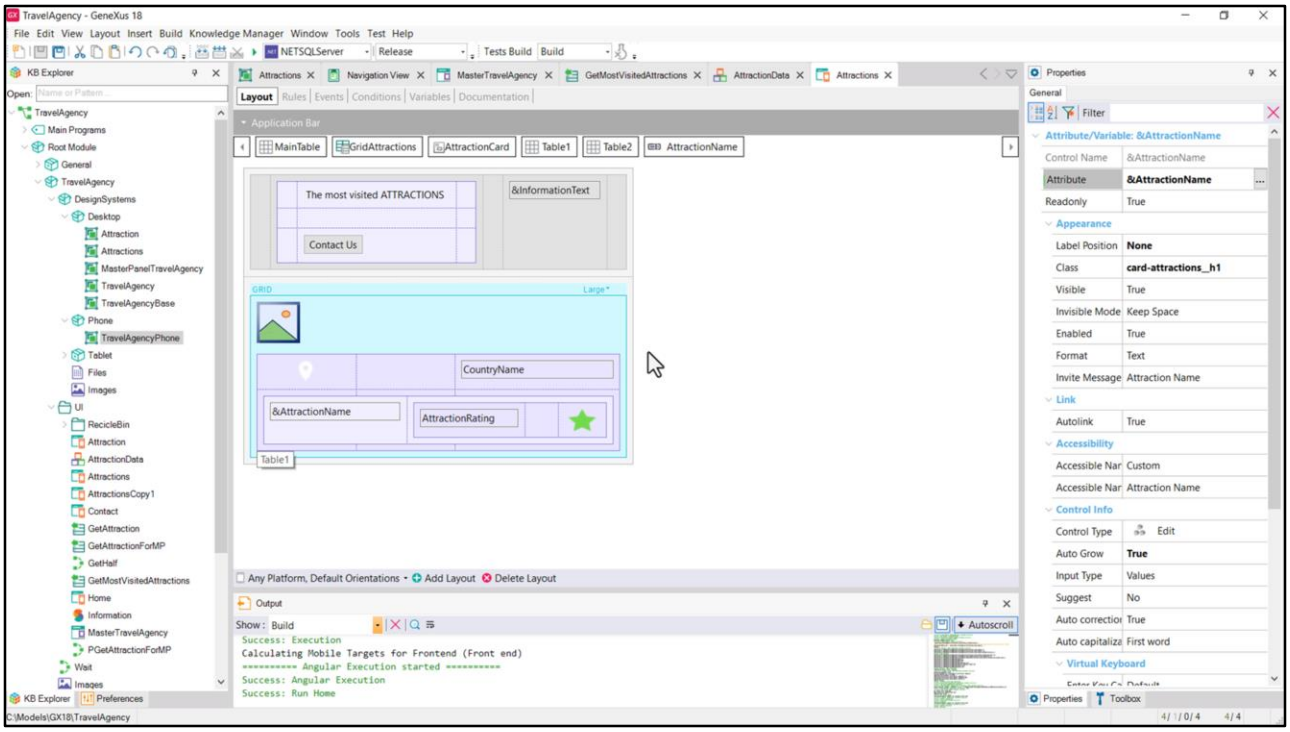
Aqueles que ficam divididos, em nossa solução com atributos, é porque o texto *em si* não cabe no espaço.



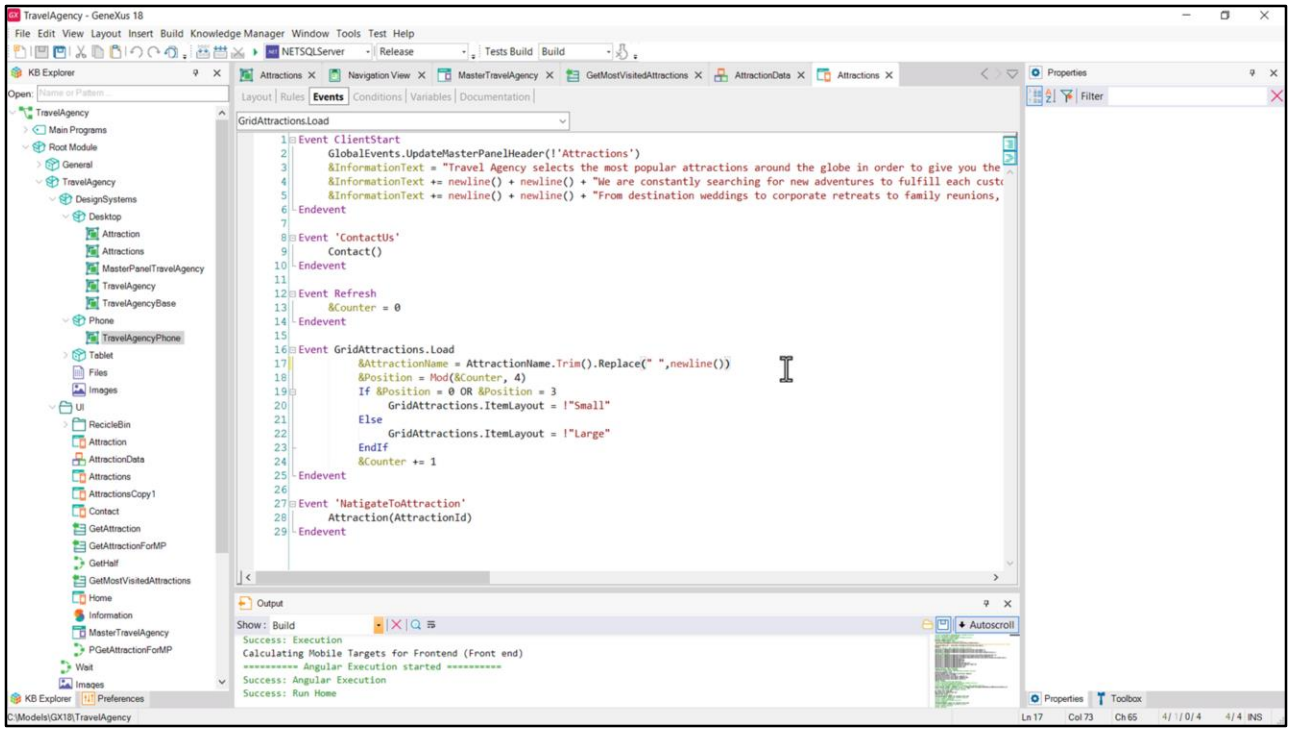
É que ao utilizar o atributo, seu conteúdo é exibido conforme está carregado na base de dados.



Na minha solução, no entanto, eu processo o que retorno ao SDT para este campo desta forma.

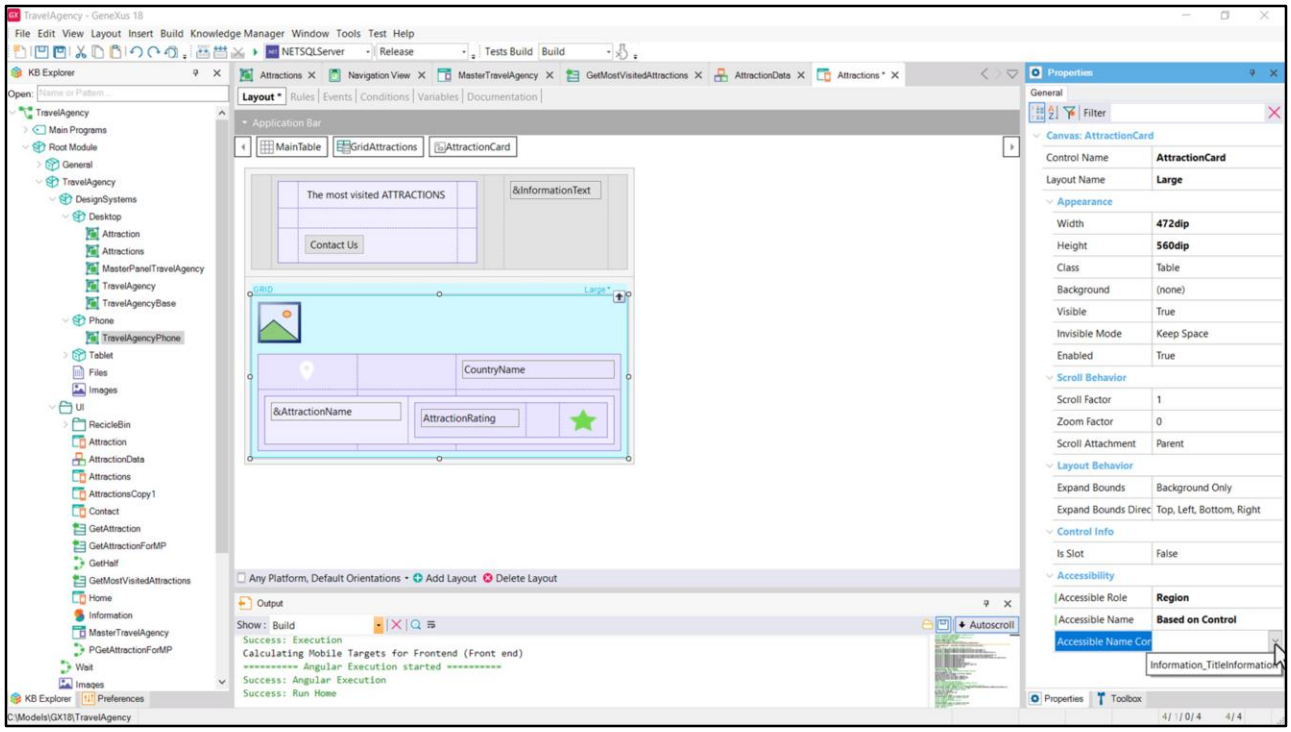


Para adaptar minha solução, teria que colocar uma variável aqui em vez do atributo, variável que me convém basear no atributo (por exemplo, chamando-a igual, automaticamente será oferecido para baseá-la no atributo de mesmo nome) e teria que fazer o mesmo para o outro item layout.

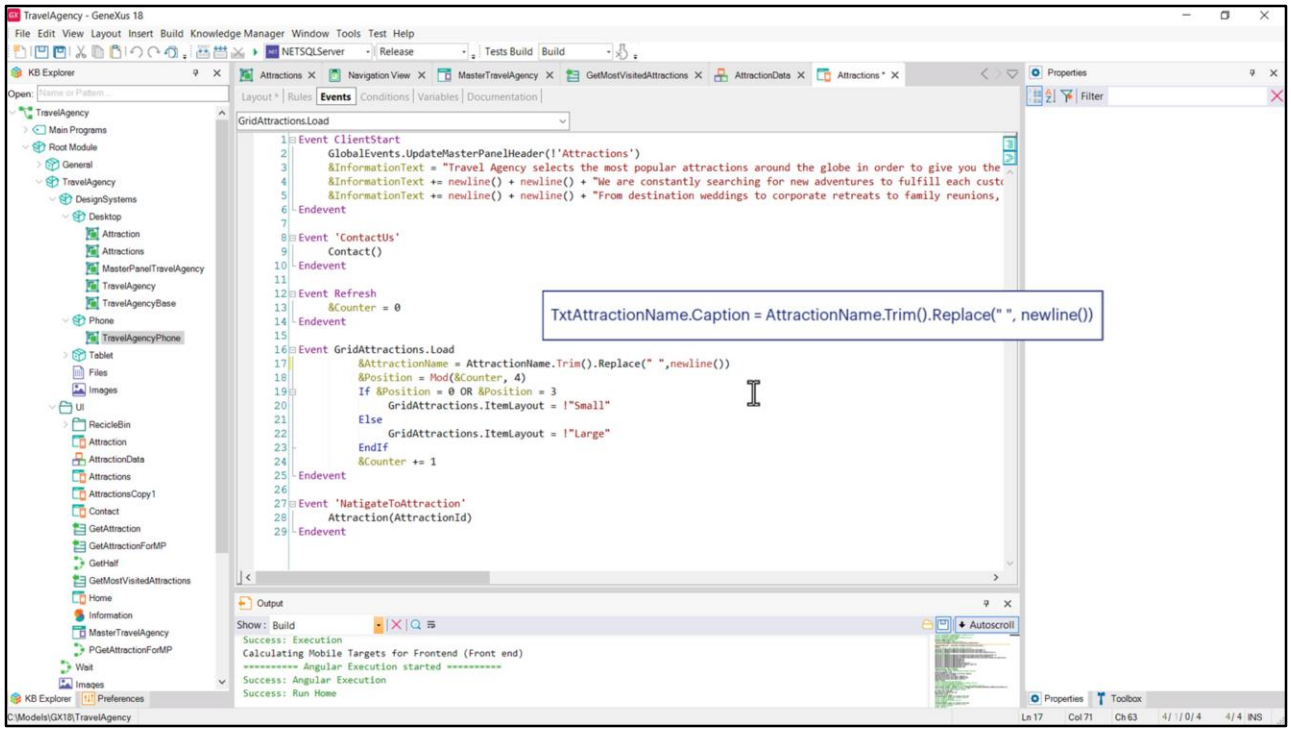


E no evento Load atribuir este valor à variável.

Se agora executarmos, vemos os textos exatamente como queríamos.

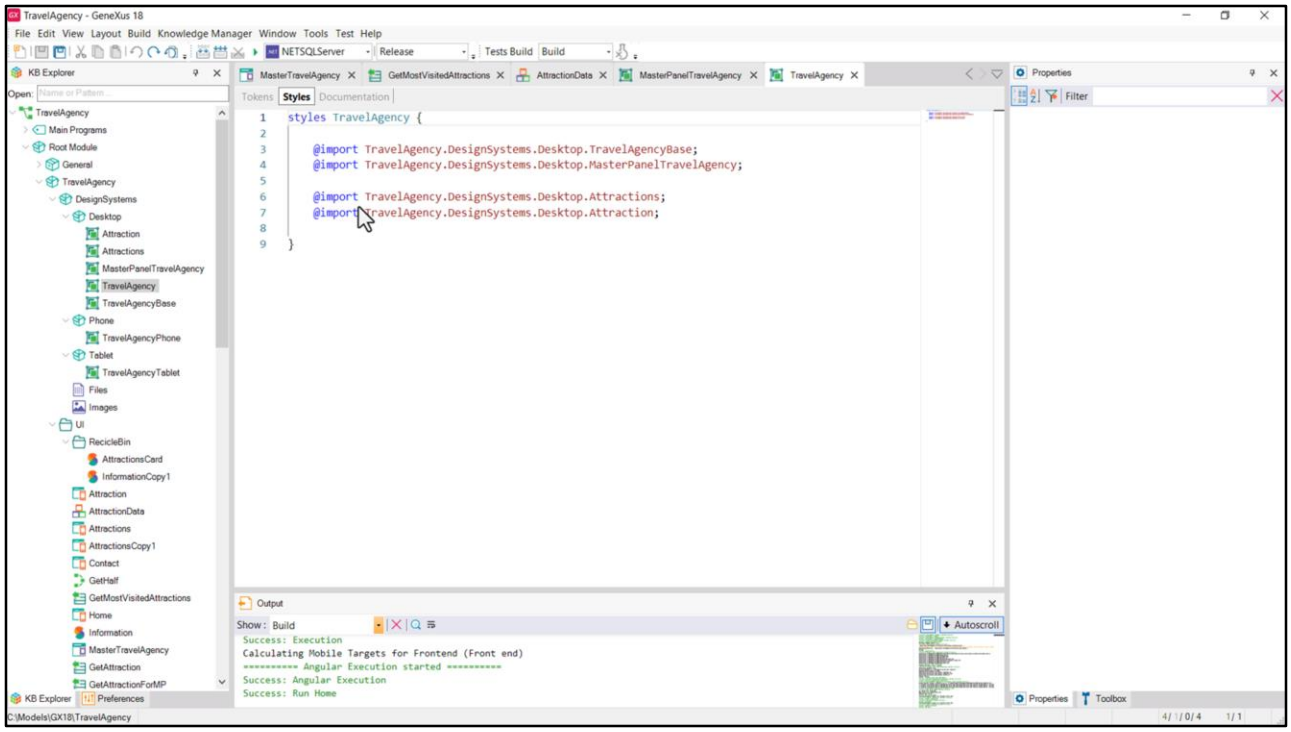


Por outro lado, para atender a acessibilidade, teríamos que fazer com que cada card mostre como nome, o nome da atração, para o que aqui deveríamos colocar Region… e em Accesible Name não Custom, mas o baseado em um controle… mas por enquanto só podemos basear em controle TextBlock e não em variável de texto… então, na verdade, em vez da variável AttractionName, deveríamos colocar um TextBlock aqui…

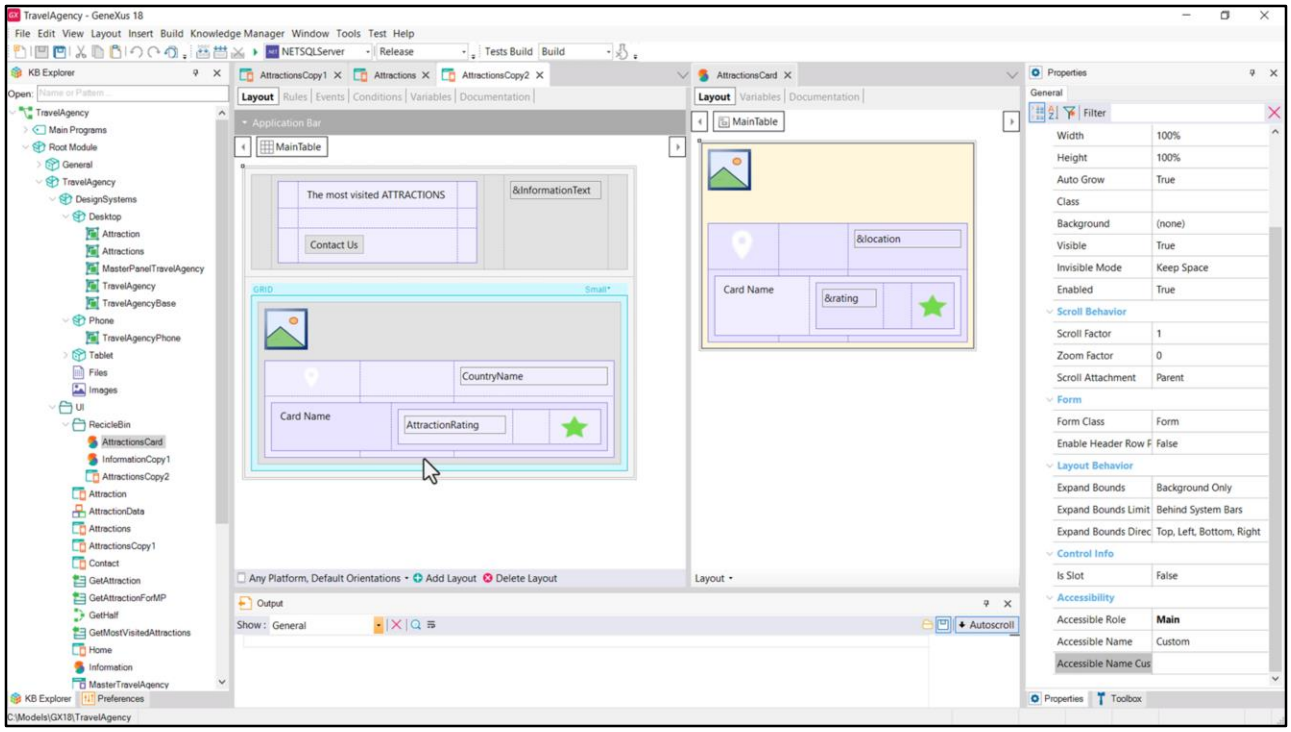


…e fazer esta atribuição, mas para a propriedade Caption desse textblock. Para não me alongar mais, deixo isso para que vocês façam.

O grid também possui propriedades de acessibilidade.



Não mencionei isso explicitamente, então faço isso agora: criei um DSO por objeto cuja UI tenho trabalhado até agora. Então tenho o do Master Panel, o de Attraction, o de Attractions que foi no qual fizemos tudo isso…
O de Base, e no TravelAgency que será o pai de todos, claro que tive que incluir todos esses outros.



Por último, poderíamos ter utilizado um stencil para não duplicar o layout dos cards, já que a única diferença entre o item Small e o Large era a altura do canvas e nada mais.

Então poderíamos ter inserido para cada item layout um controle stencil... dessa forma, tudo relacionado ao layout do card é resolvido apenas uma vez. Por exemplo, se quiséssemos alterar a distância desta tabela em relação à Top. Fazemos isso aqui e afetará automaticamente ambos os item layouts.

No entanto, há algumas especificações a serem feitas para esta solução, que têm a ver com como se trabalha com valores relativos para a tabela interna do grid, nas quais agora, não podemos entrar, e por isso não vou aprofundar nesta solução. Mas deixo um xpz com esses objetos, onde poderão investigar tudo isso.


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