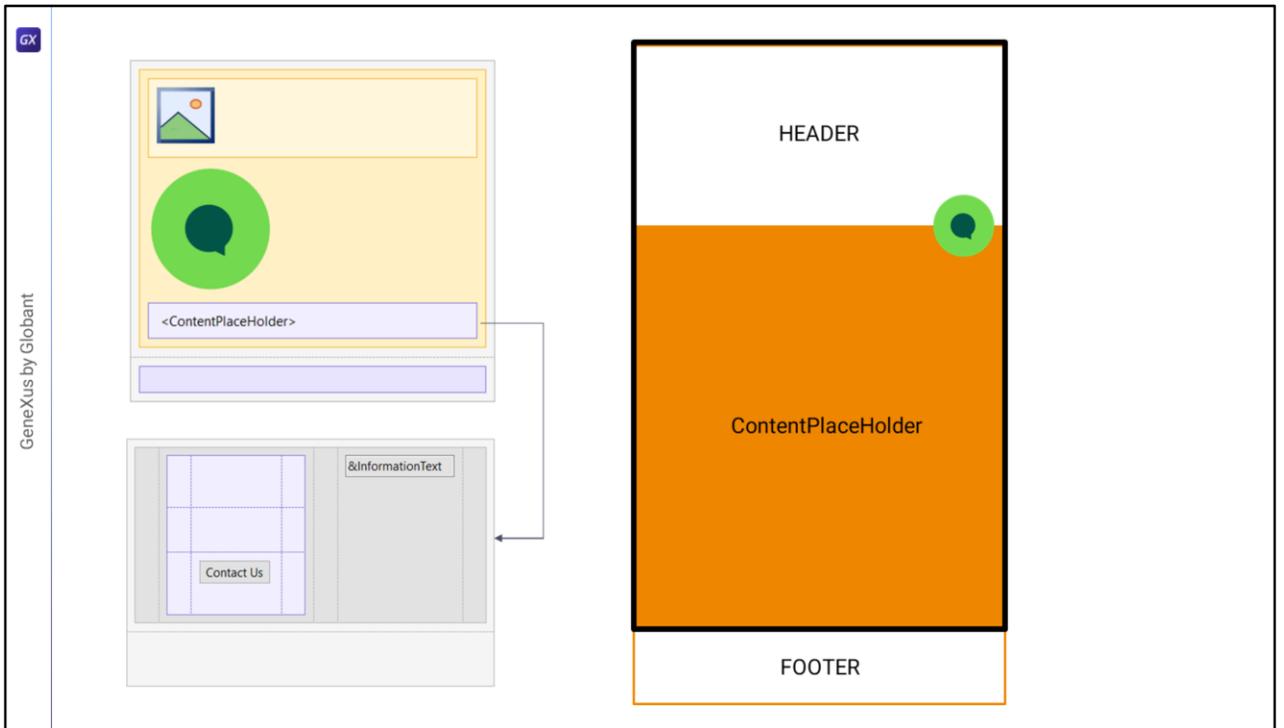


Header in GeneXus: background image and chatbot button

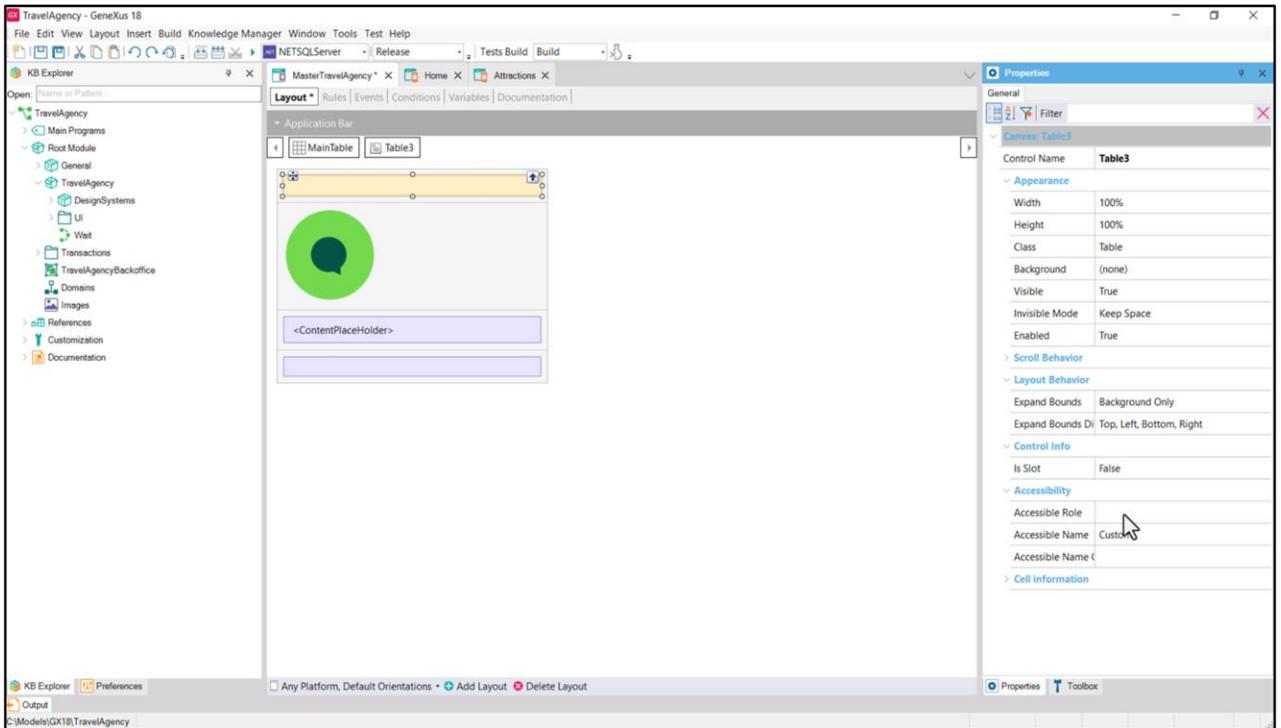


Cecilia Fernández

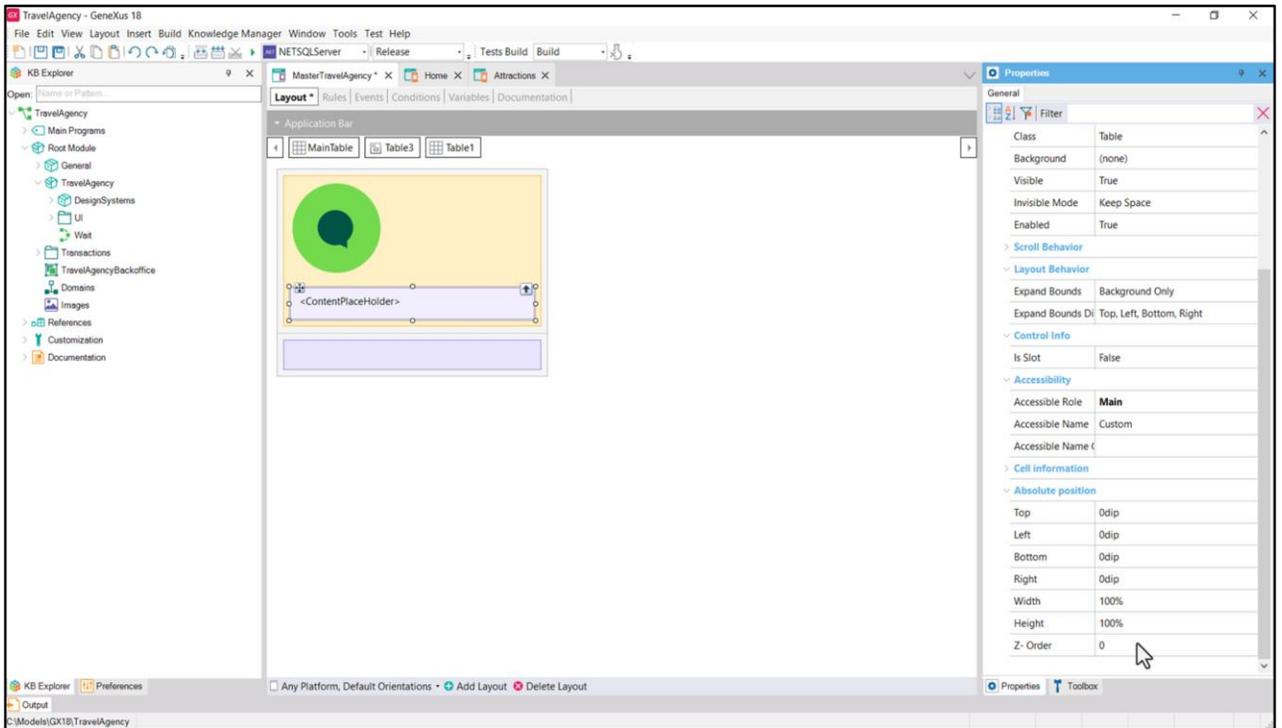


No vídeo anterior analisamos no nível conceitual duas possíveis implementações para o Master Panel.

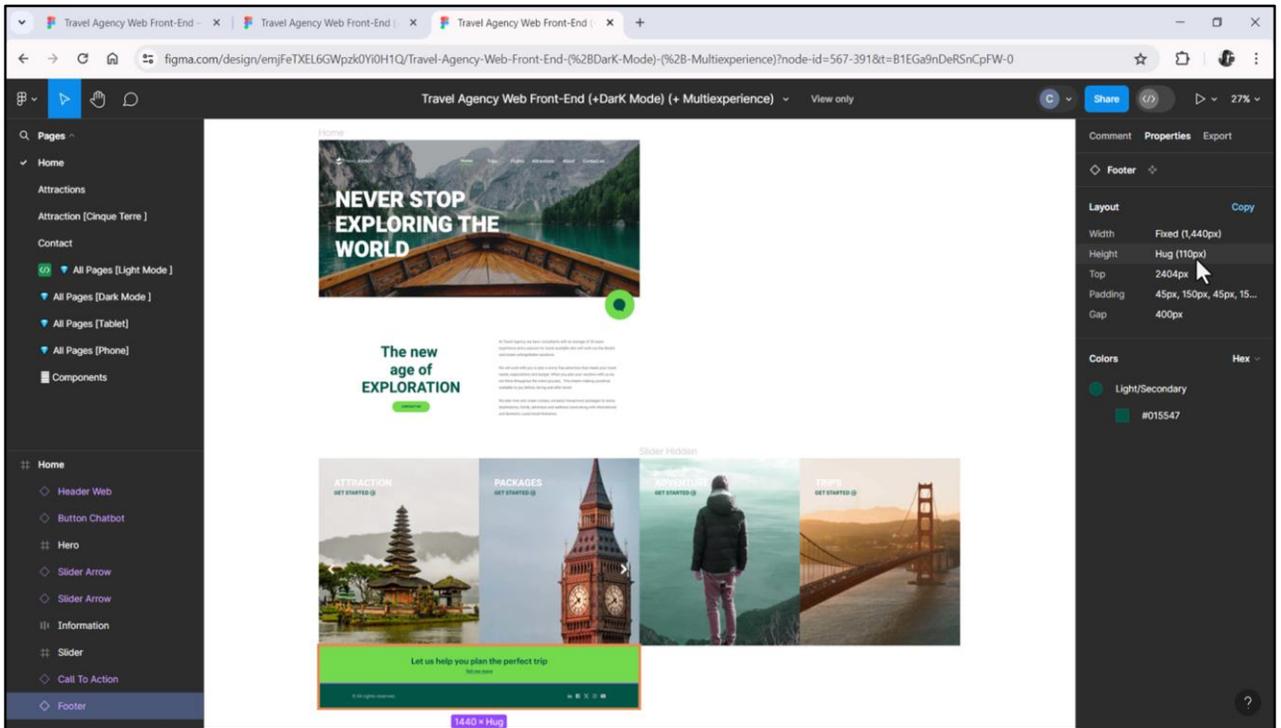
Vamos implementar a segunda solução, que foi aquela que nos pareceu conceitualmente a mais adequada.



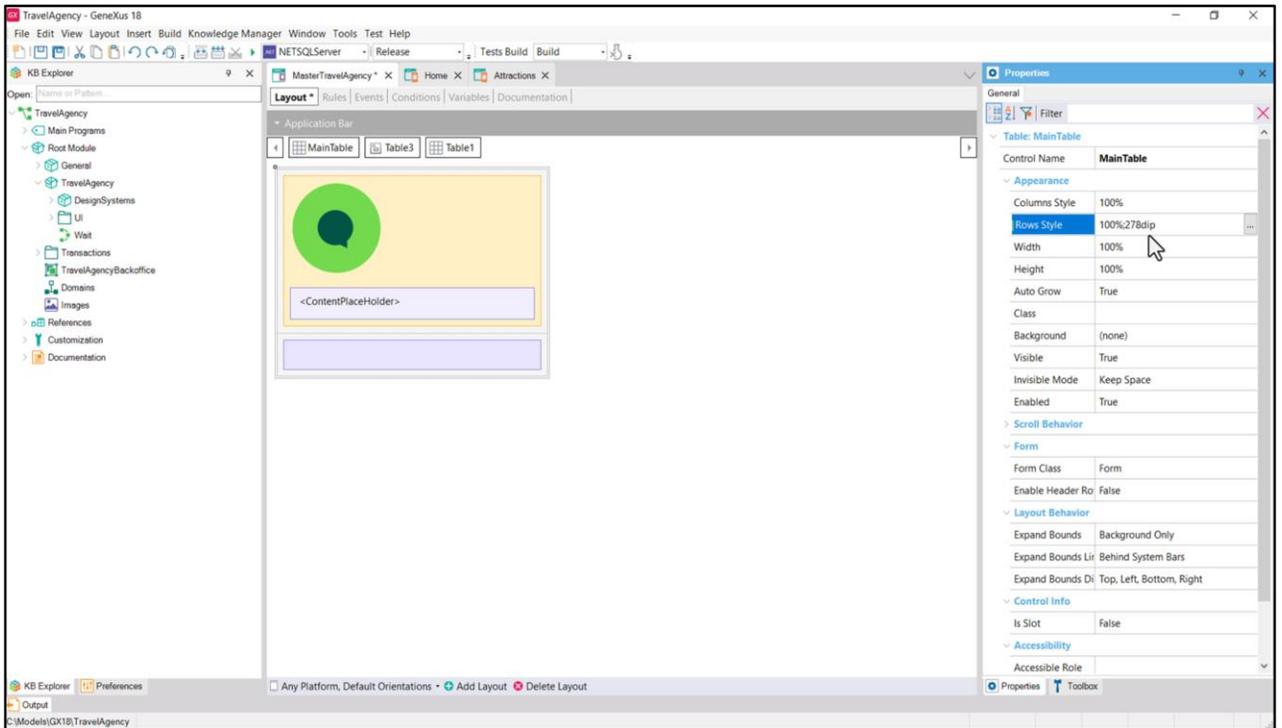
Aqui temos a tabela Main. Teremos que inserir um controle canvas na primeira linha, vamos ver suas propriedades. A esta não vamos colocar Role porque terá que agrupar o Header e o ContentPlaceHolder.



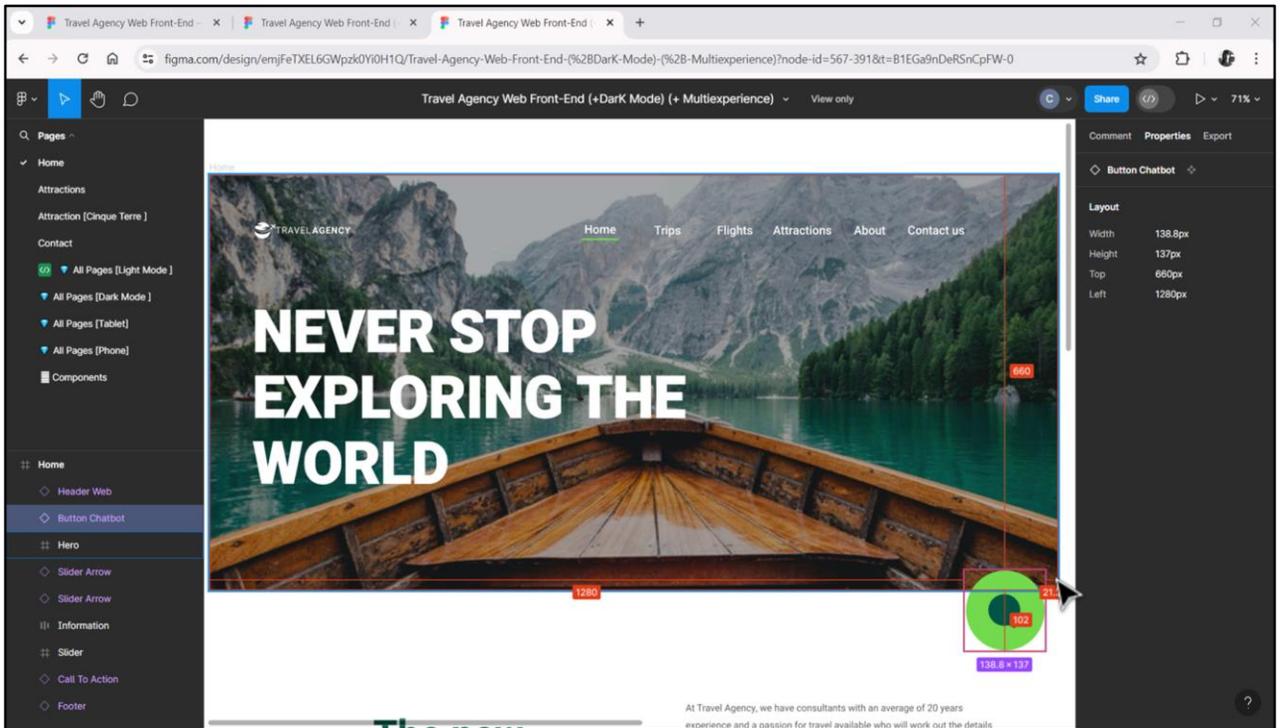
Vamos arrastar para dentro, tanto o botão do chatbot, e vemos como aparecem agora as propriedades Absolute Position...
Como a tabela com o contentplaceholder, e vemos que também aparecem as propriedades Absolute Position...



Agora a tabela Main ficou com duas linhas. A do Footer deve ter uma altura de $168 + 110$, ou seja, de 278 dips.

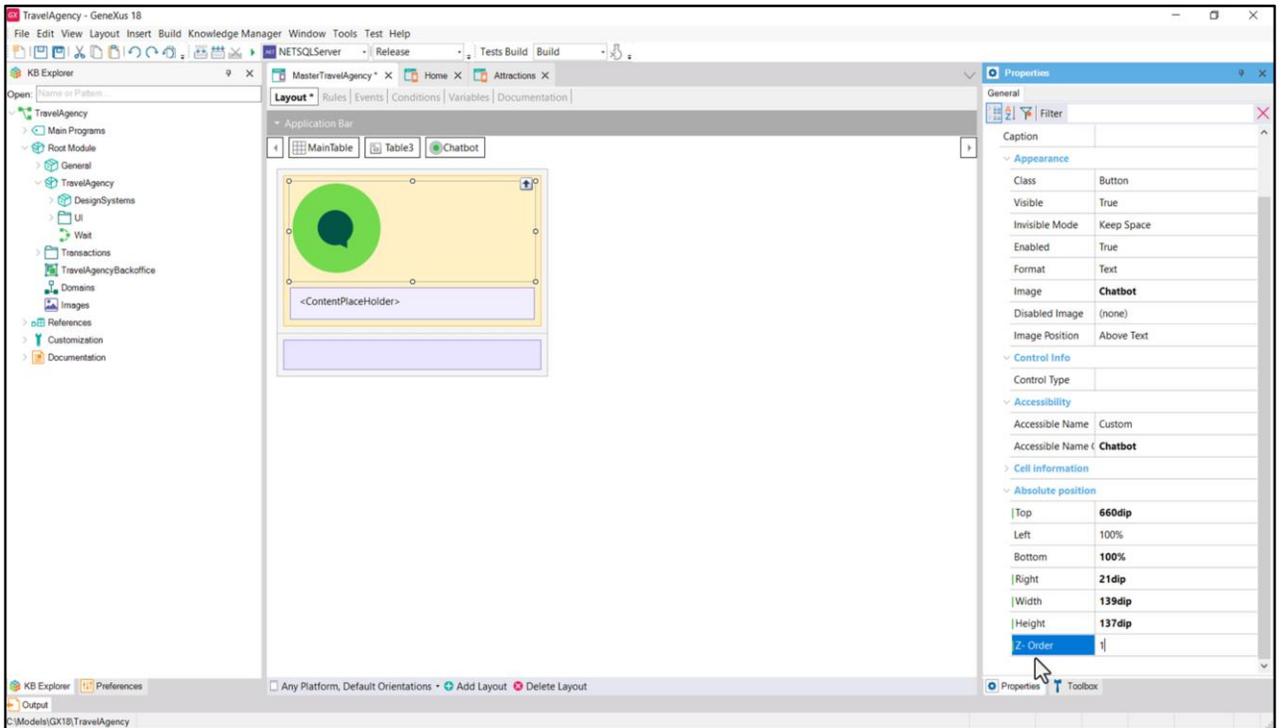


Então, para a Main table, colocamos em Rows Style: 100% para a primeira linha e 278 dips para a segunda.



Vamos dar as posições absolutas aos dois controles que temos no momento.

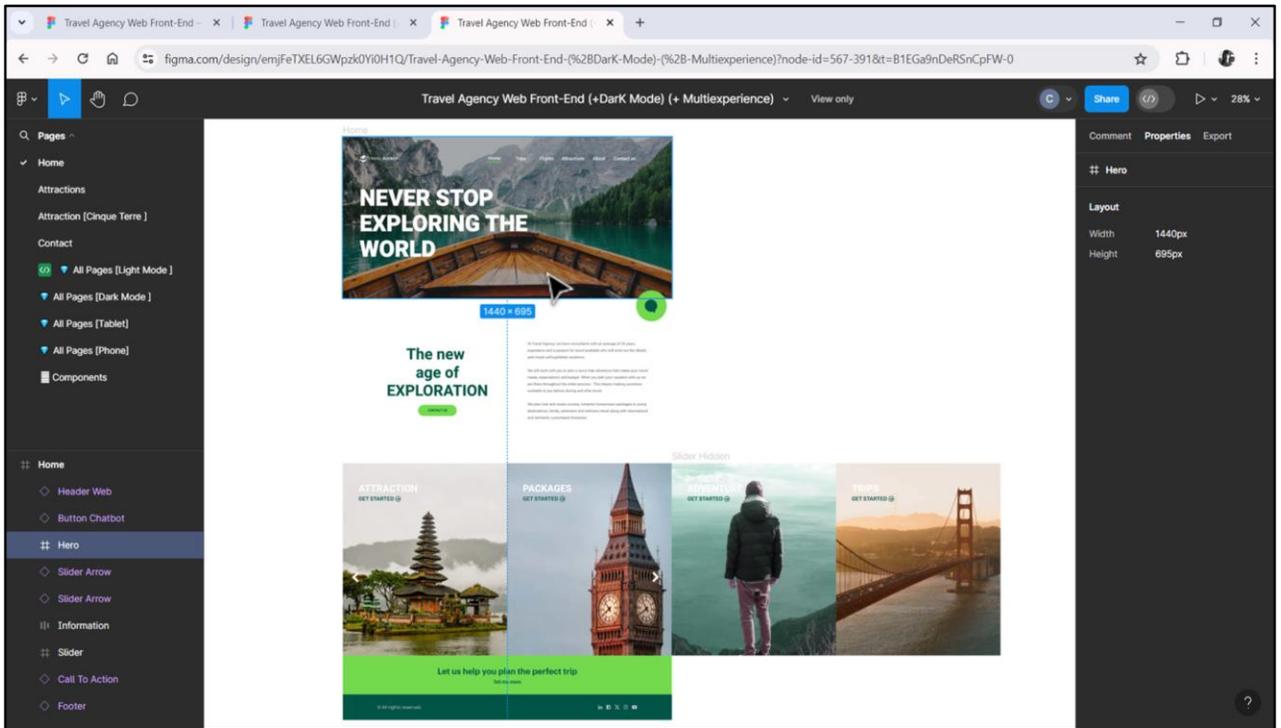
Do botão do chatbot, tínhamos que sua largura era de 139, e estava a 21 da direita...
E sua altura era de 137 e estava a 660 de cima. Então...



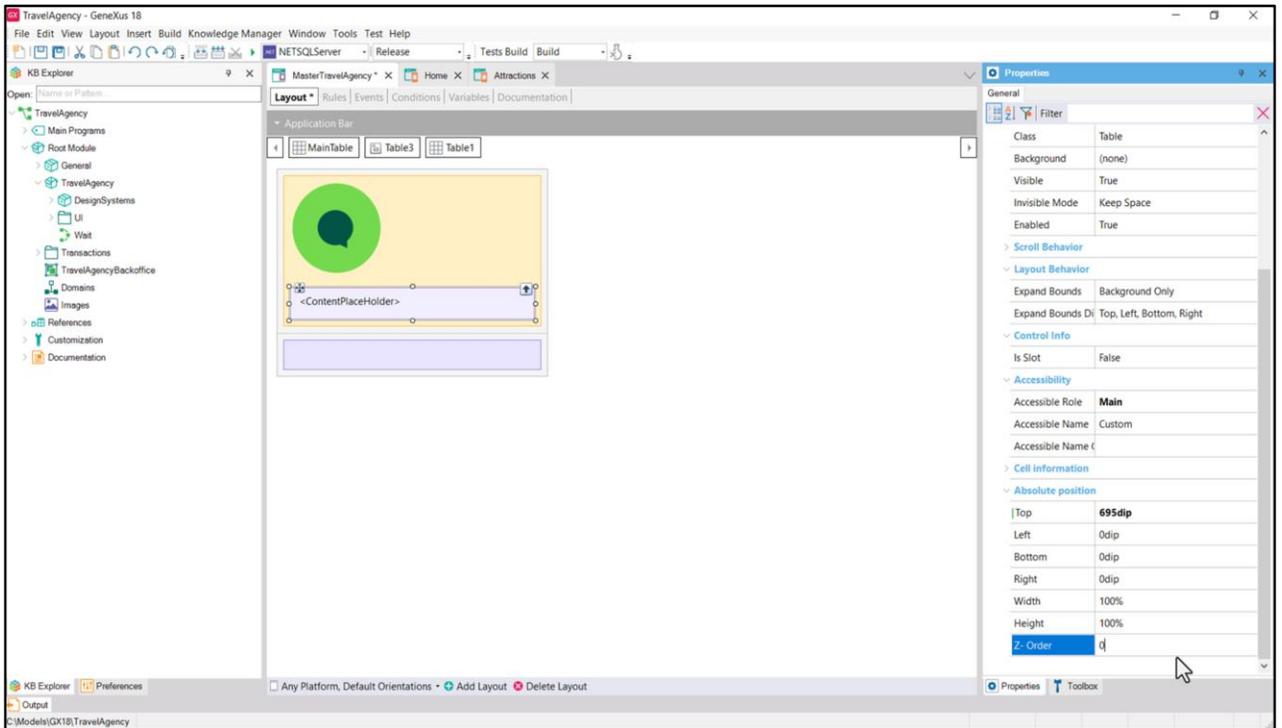
Vejamos que se fixarmos a Right em 21 e Width em 139, automaticamente a Left fica em 100%.

E da mesma forma, se fixarmos a Top em 660 e Height em 137, Bottom fica em 100%.

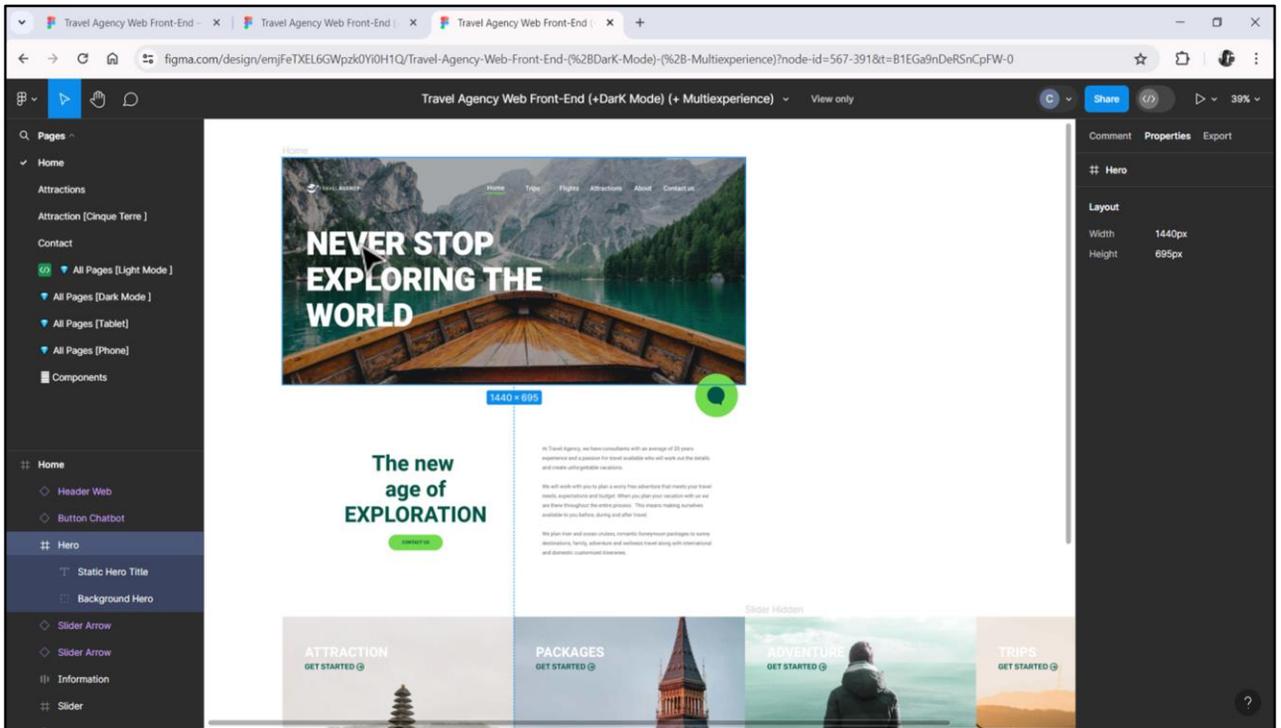
Como este botão é o que deve ficar na camada mais acima, colocaremos 1 para a Z-order, porque o contentplaceholder e o Header (que ainda não inserimos) podem estar na mesma camada 0, a inferior.



Agora precisamos especificar o posicionamento absoluto da tabela com o contentplaceholder. Sabemos que deve começar aqui, ou seja, a 695 dips do Top do canvas... e que se cole ao restante das bordas do canvas.

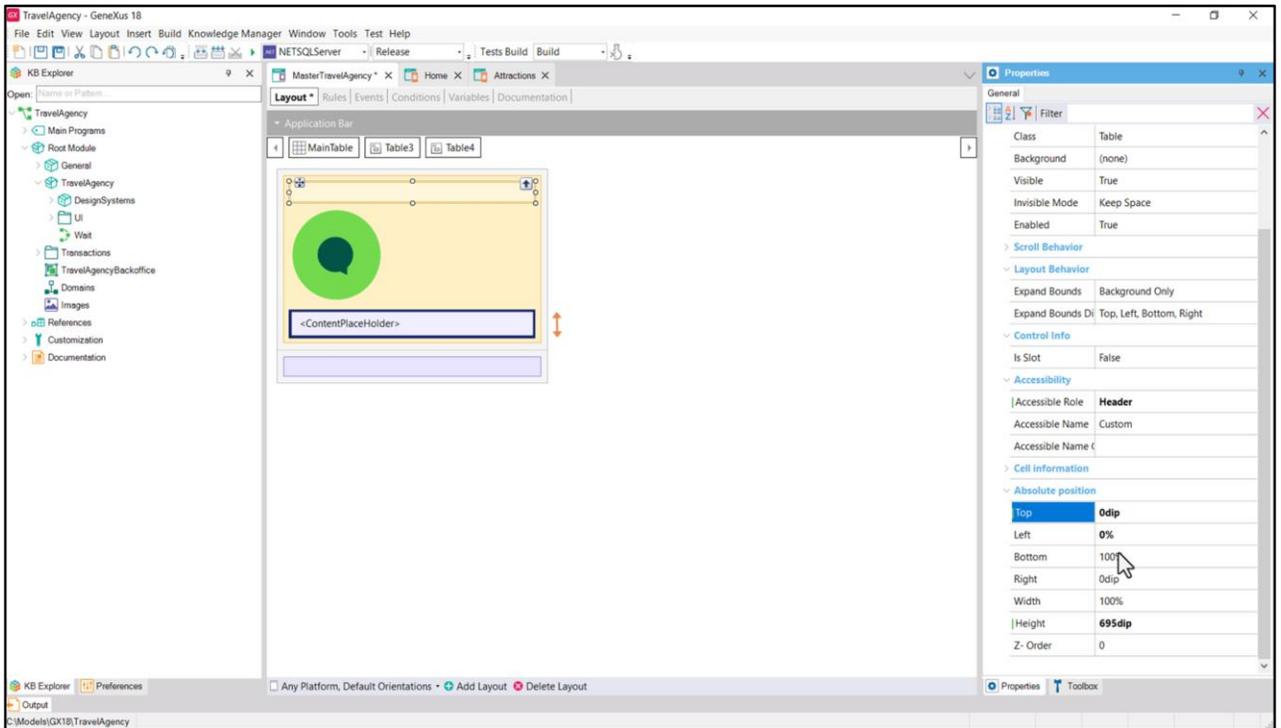


Ou seja: Top 695, e o restante tal como está: 0 da esquerda, 100% de largura e 0 da direita. 100% de altura e 0 de Bottom. E Z-order também 0.



Agora falta implementar o Header propriamente dito, que conterà tanto a imagem de fundo, como o menu, o logo e este texto.

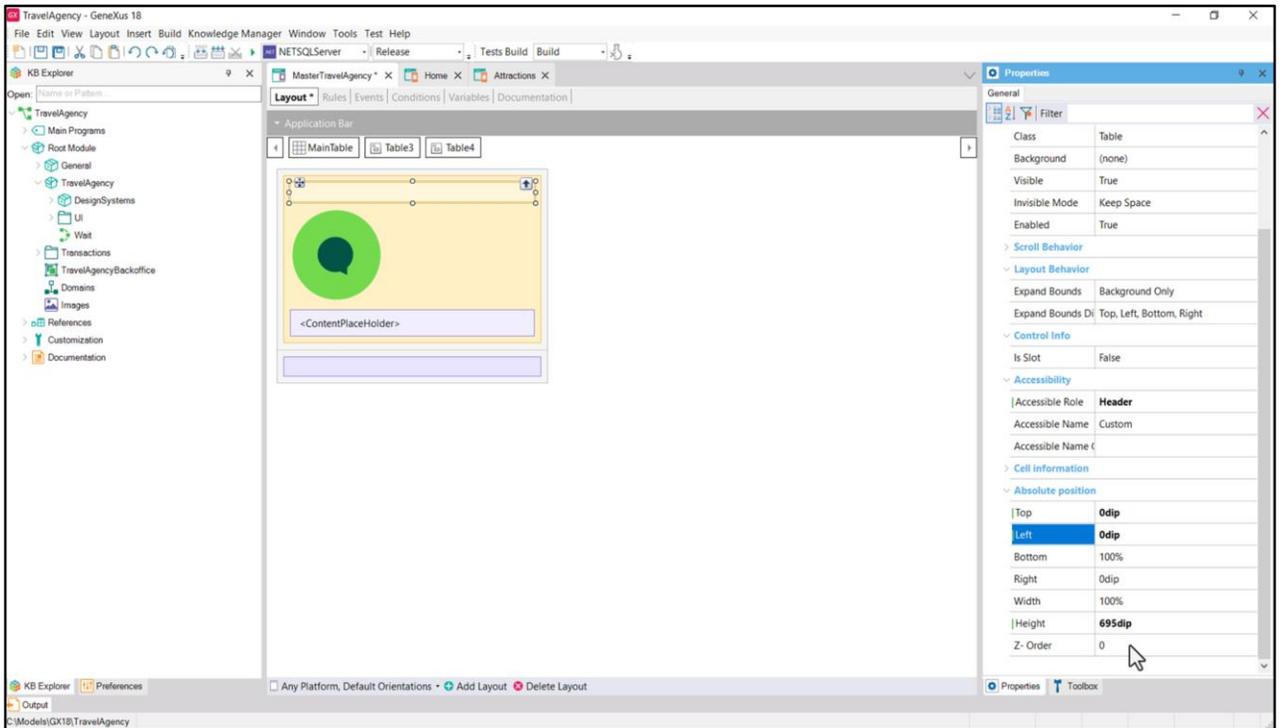
Precisaremos de outro canvas para sobrepor tudo isso. Não pode ser o mesmo onde estão o botão e a tabela com o contentplaceholder porque precisamos atribuir Role Header a ele para acessibilidade. Então precisamos de um container separado.



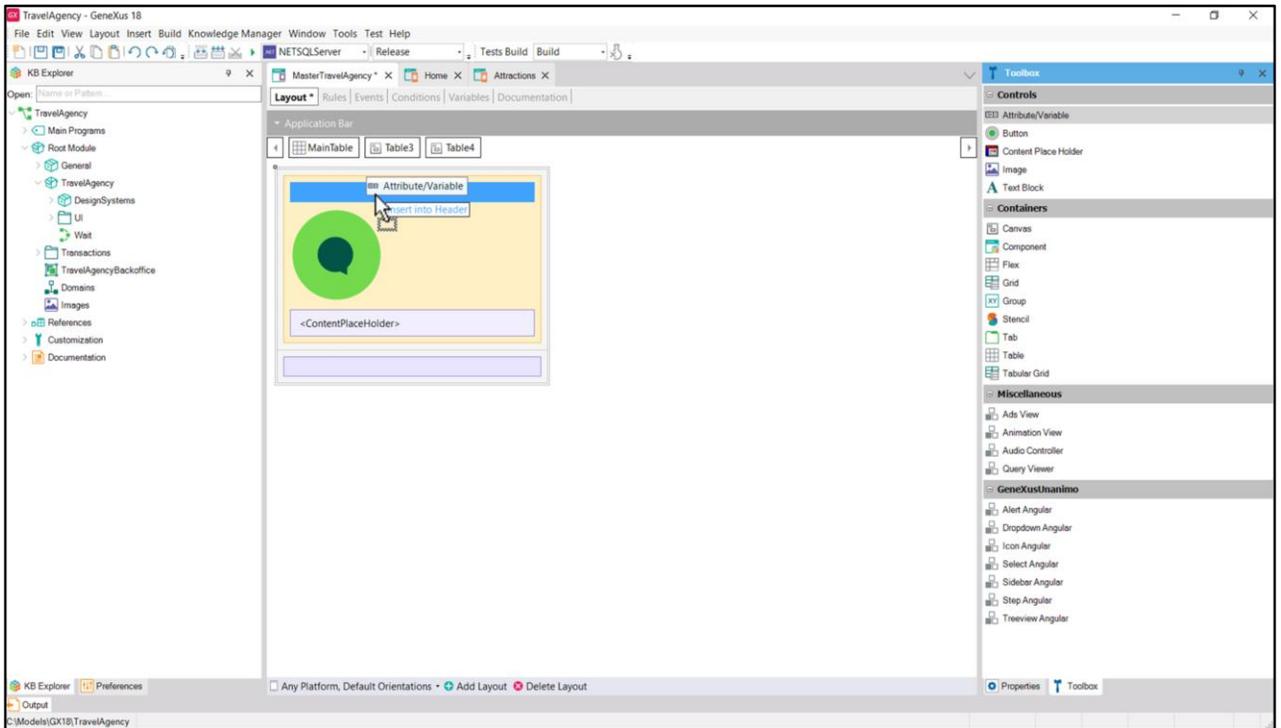
Inserimos então, outro Canvas, que chamaremos de Header e colocaremos em Accessible Role o valor Header.

E qual será seu posicionamento absoluto em relação ao canvas externo?

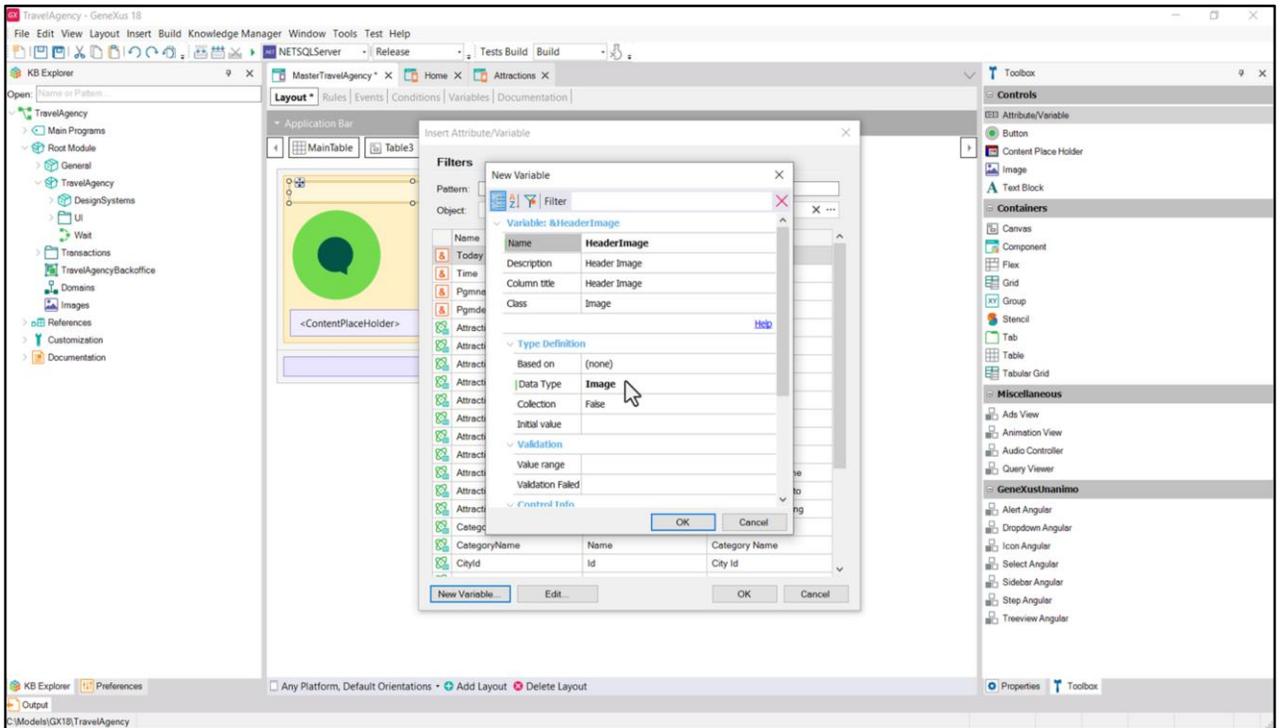
Primeiramente, sua altura será de 695 dips, que era a altura da imagem de fundo. Queremos que fique colado ao Top, então 0 dips de cima, o que o deixa de Bottom 100% restante (que corresponderá à altura do contentplaceholder).



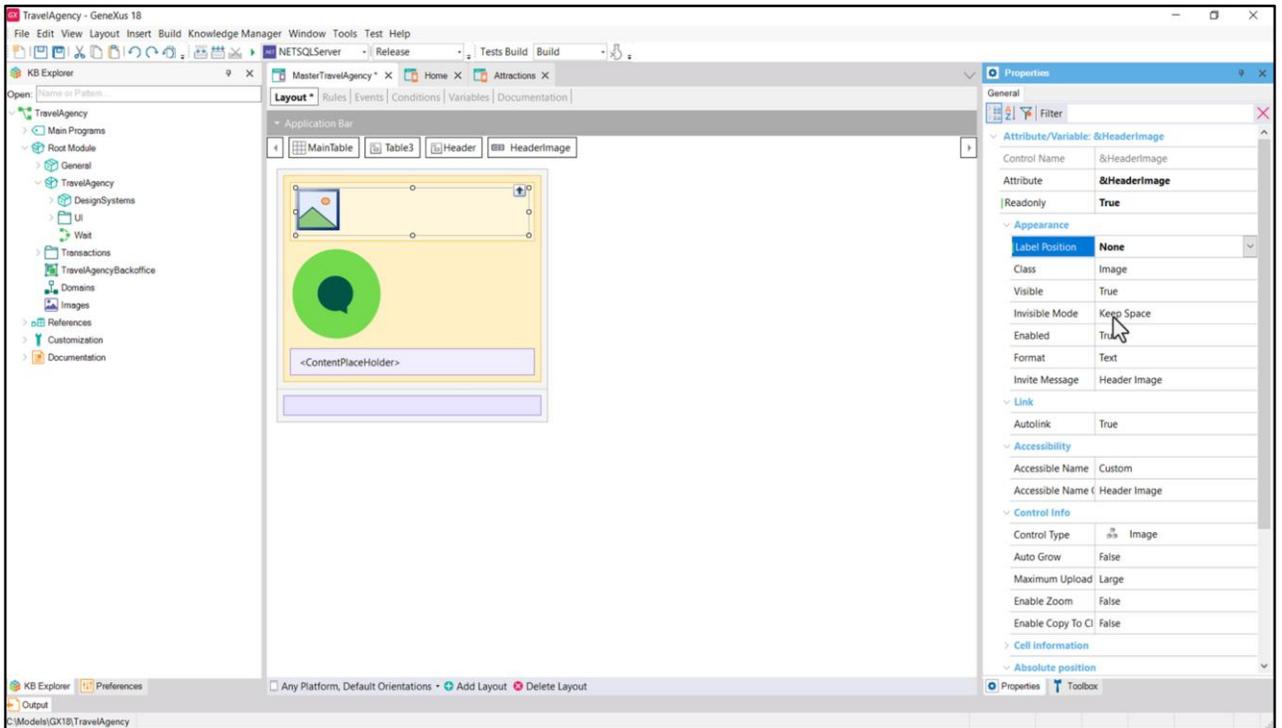
E da esquerda e direita ficará também a 0 dips, ou seja, colado, já que sua largura será de 100%. A propriedade Z-order, como já analisamos, ficará com o valor 0.



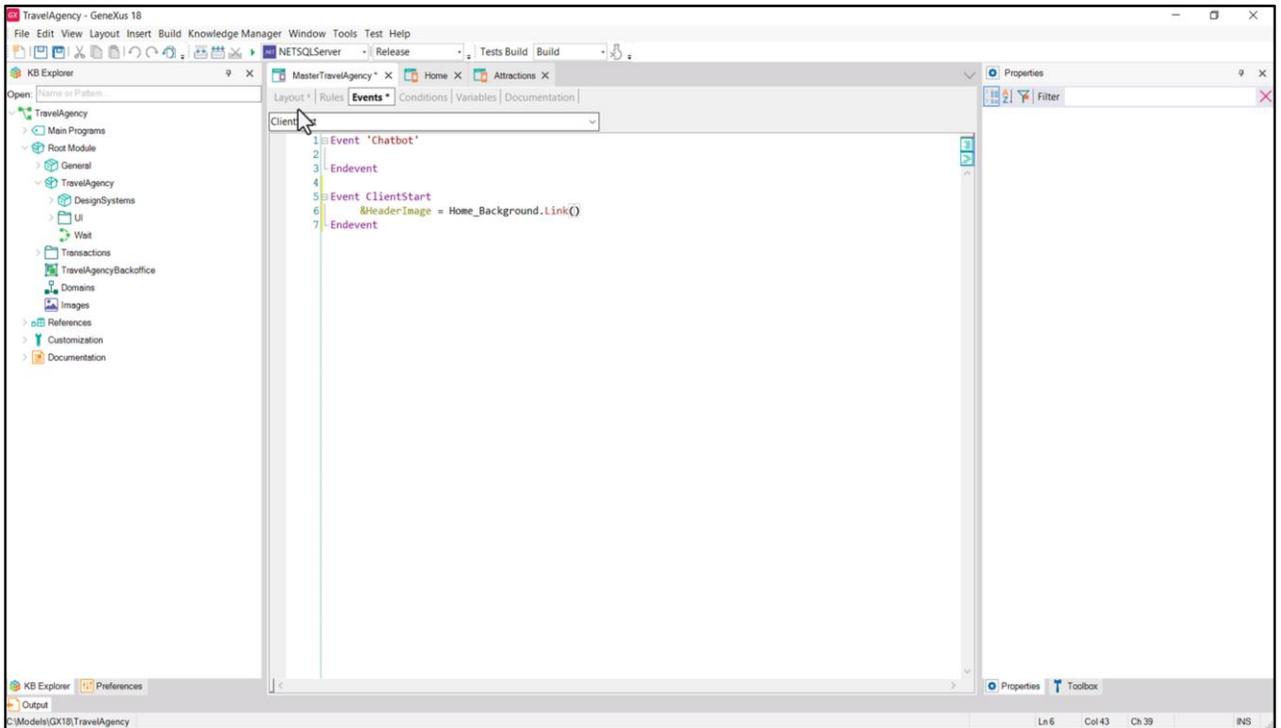
Agora preciso implementar a imagem de fundo, aquela que normalmente chamamos de Hero. Tenho duas opções: ou utilizo um controle Image ou utilizo um controle variable que contenha a imagem. Utilizarei a segunda alternativa, pois essa imagem irá variar dependendo do panel que estiver sendo carregado no Contentplaceholder em cada oportunidade.



Vou chamar de HeaderImage e será do tipo de dados Image.

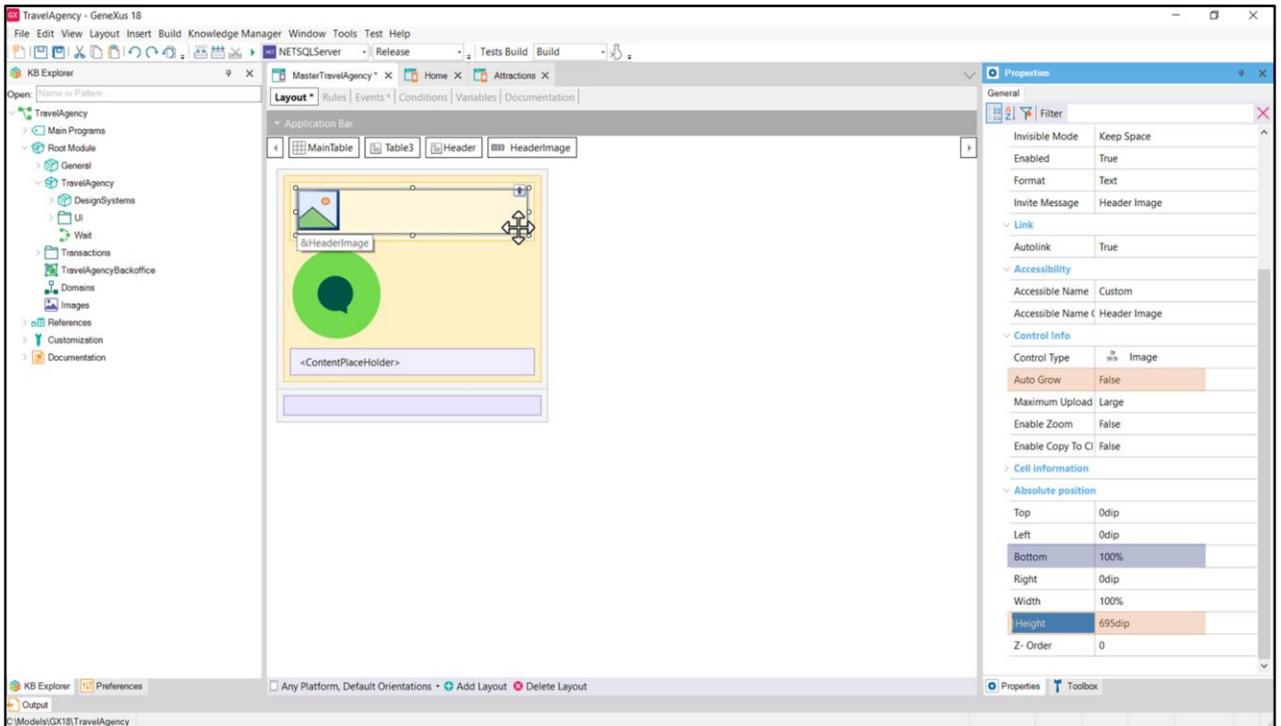


Quero que seja readonly e que não mostre seu rótulo.



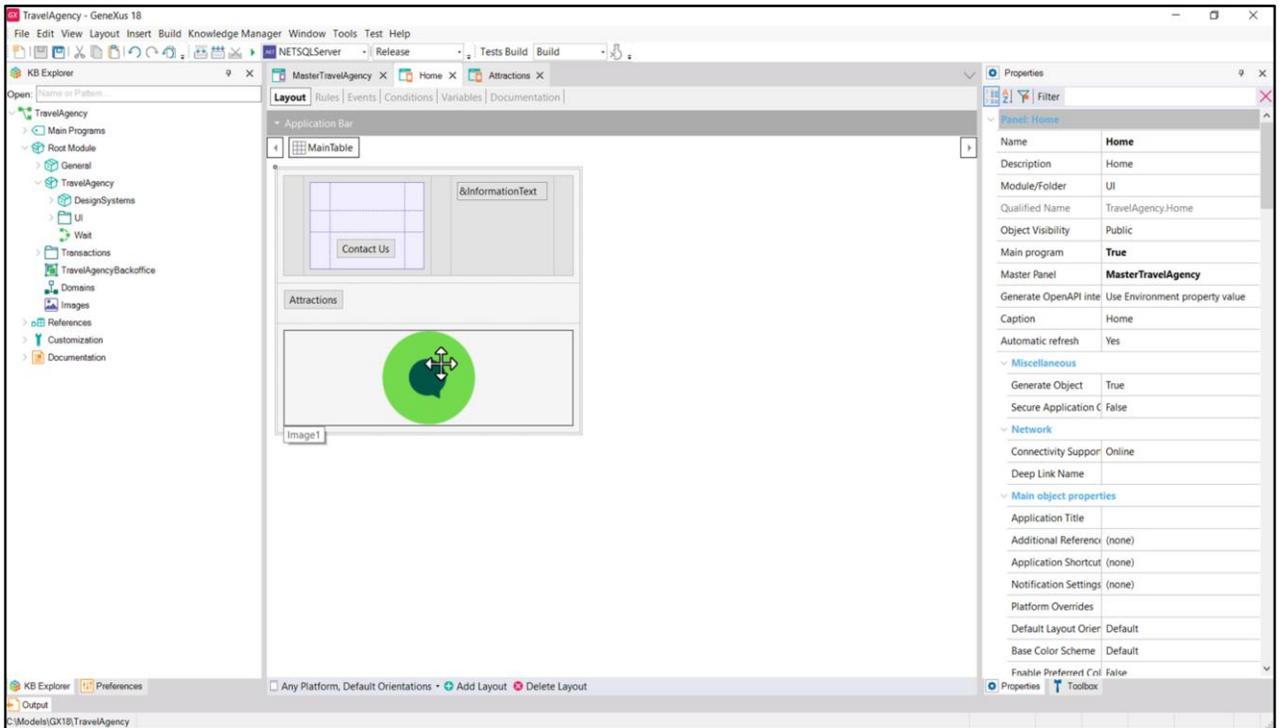
Vou carregá-la provisoriamente no evento ClientStart, a partir (ctrl-o) do objeto imagem que já havíamos inserido na KB na fase de preparação. E que havíamos chamado assim... Utilizo o método Link.

Depois precisaremos fazer variar essa atribuição dependendo de quem estiver sendo carregado, mas veremos isso mais adiante. Agora vamos deixá-la fixa.

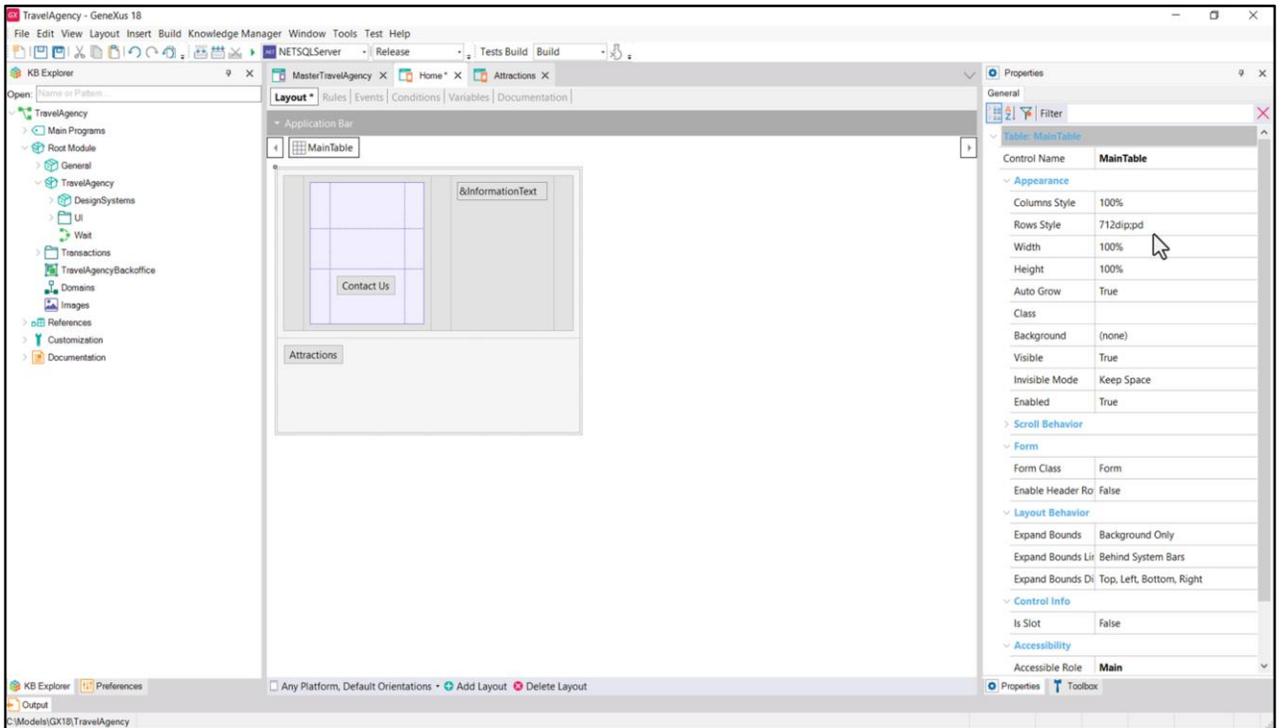


O próximo passo será indicar o posicionamento absoluto da variável em relação ao Canvas Header. Diremos, então, que sua altura será de 695 dips. E deixaremos em False o Auto Grow. As outras propriedades deixamos como estão, porque queremos que a imagem fique colada ao top e às bordas laterais. O Canvas, como dissemos no vídeo anterior, tem internamente Auto Grow em true, então qualquer controle que transborde fará com que ele se expanda para baixo. Mas neste caso temos apenas a imagem, e fixada a 695 dips, então este bottom de 100%, quando for calculado no carregamento da página, será de 0 dips. Ou seja, a imagem deve ser colada ao canvas nos quatro lados.

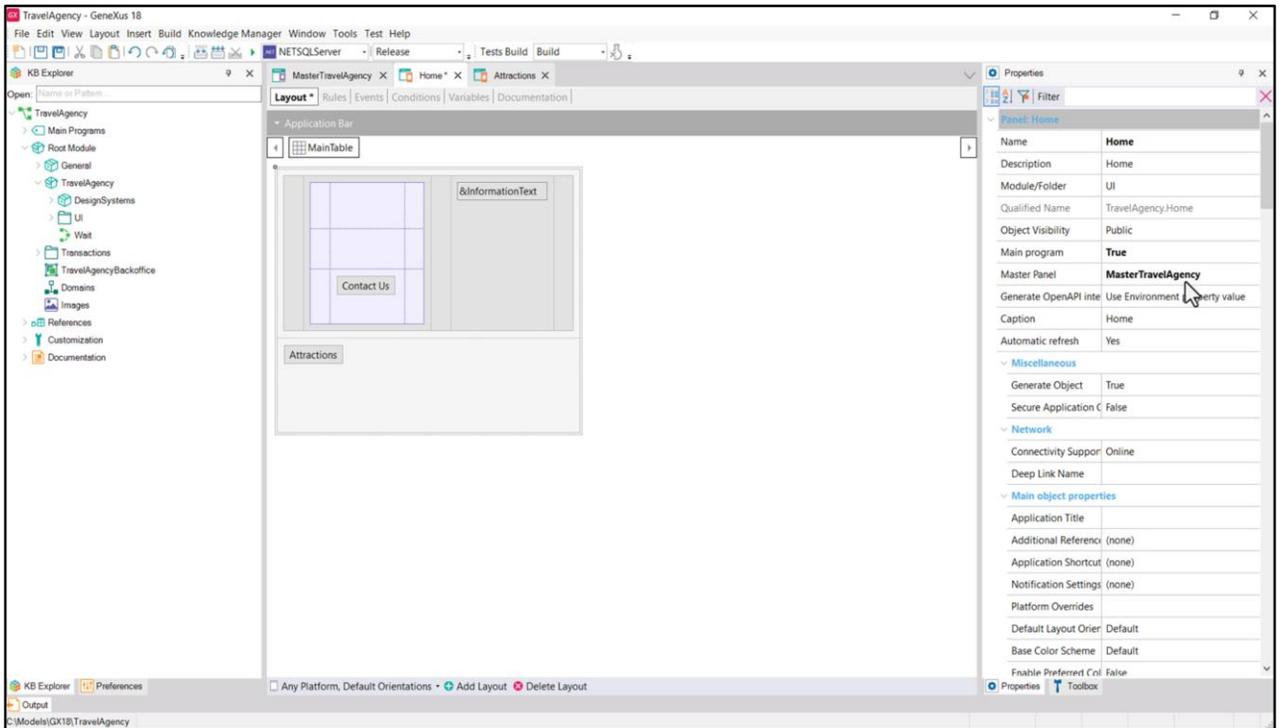
Vamos gravar o Master Panel.



No Panel Home colocamos esta imagem para mostrar o que não lembro mais. Nós a removemos.
O botão para chamar Attractions, por enquanto deixamos.

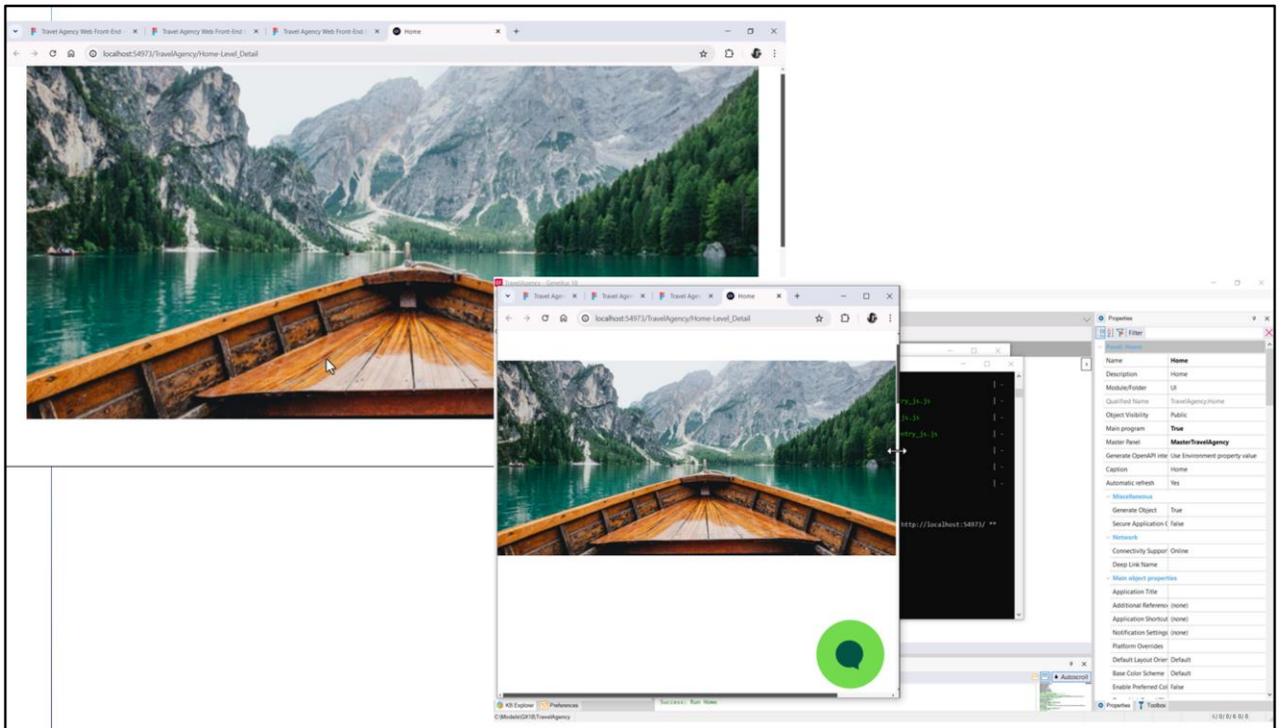


E Rows Style continua como a tínhamos antes. Com esta solução não precisamos fazer nenhuma alteração aqui.



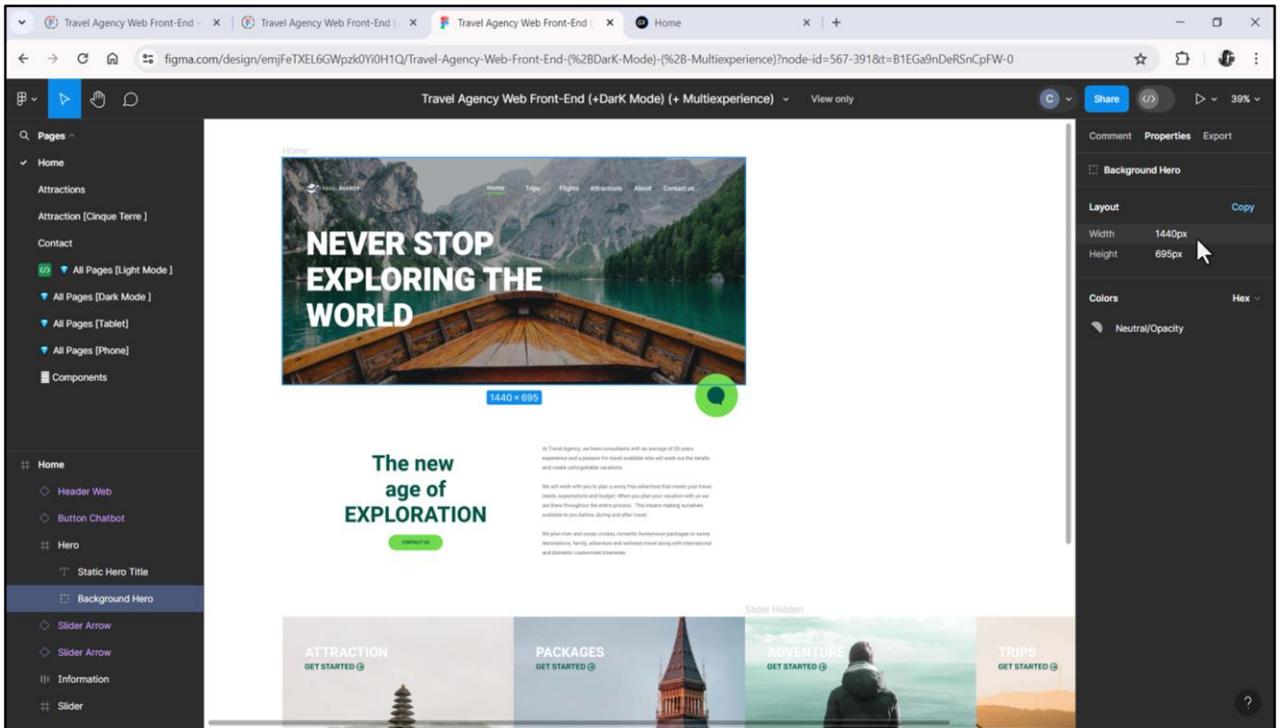
Vemos que Home tem como Master Panel aquele que estvamos trabalhando.

Vamos gravar e executar este Home.

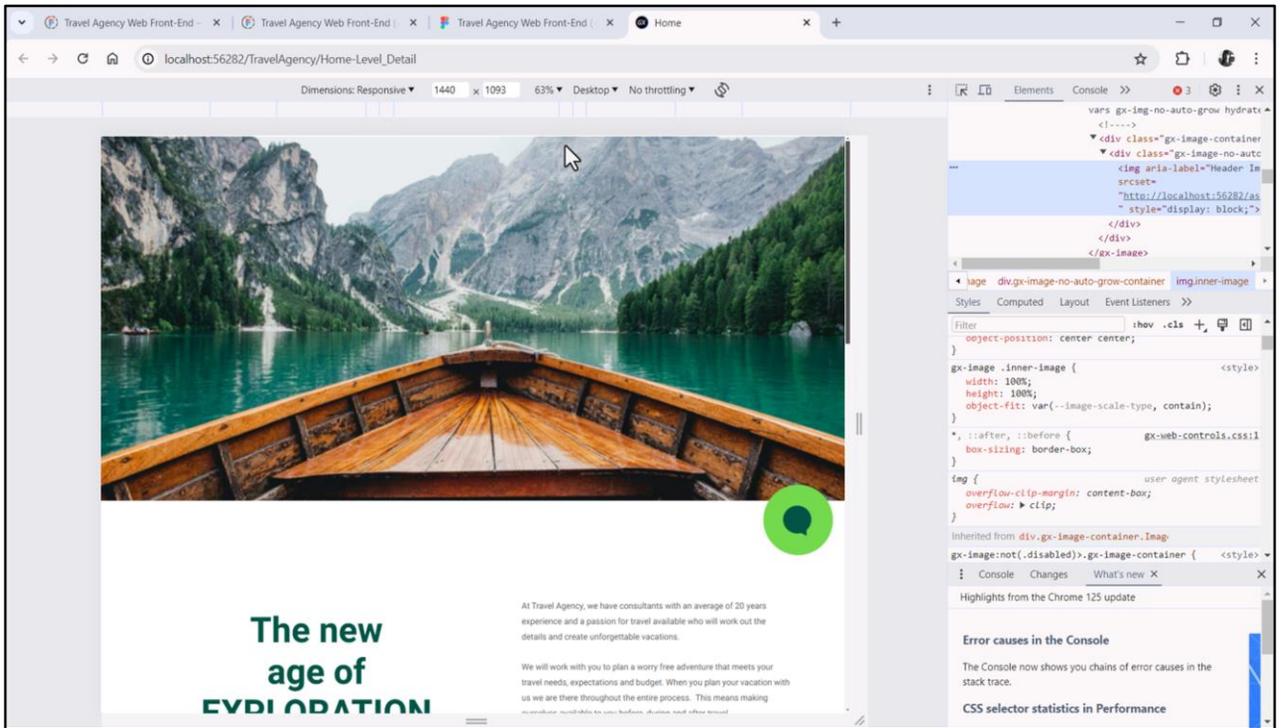


Aqui vemos o panel Home, mas algo não está certo com a imagem. Não está colada nas bordas.

E se, por exemplo, comprimirmos a largura do navegador, vemos que a imagem encolhe para ser exibida sempre por completo. Por outro lado, o botão do chatbot está sempre onde dissemos, a uma distância de 660 dips da posição top. Mas e a imagem? Vejam o que está fazendo.



A imagem que inserimos na KB tem de largura 1440 pixels e de altura 695, porque a exportamos do Figma, lembram? Com suas 3 densidades.



Se a inspecionarmos no Chrome, veremos que quando a largura de tela é de 1440, somente aí ela fica do jeito que queremos.

Travel Agency Web Front-End x Travel Agency Web Front-End x Travel Agency Web Front-End x Home

localhost:56282/TravelAgency/Home-Level_Detail

Dimensions: Responsive 1746 x 1093 63% Desktop No throttling



The new age of EXPLORATION

All Travel Agency, we have consultants with an average of 20 years experience and a passion for travel available who will work out the details and create unforgettable vacations.

We will work with you to plan a worry free adventure that meets your travel needs, expectations and budget. When you plan your vacation with us we are there throughout the entire process. This means making ourselves available to you before, during and after travel.

```
vars gx-img-no-auto-grow hydrat<
<!-->
<div class="gx-image-container"
  <div class="gx-image-no-autc
    <img aria-label="Header Im
      srcset="
        "http://localhost:56282/as
        " style="display: block;"
    </div>
  </div>
</gx-image>
```

lage div.gx-image-no-auto-grow-container img.inner-image

Styles Computed Layout Event Listeners >>

Filter

object-position: center center;

```
gx-image .inner-image { <style>
width: 100%;
height: 100%;
object-fit: var(--image-scale-type, contain);
}
::after, ::before {
box-sizing: border-box;
}
img {
overflow: clip-margin: content-box;
overflow: clip;
}
Inherited from div.gx-image-container.Img:
gx-image:not(.disabled).gx-image-container { <style>
```

Console Changes What's new X

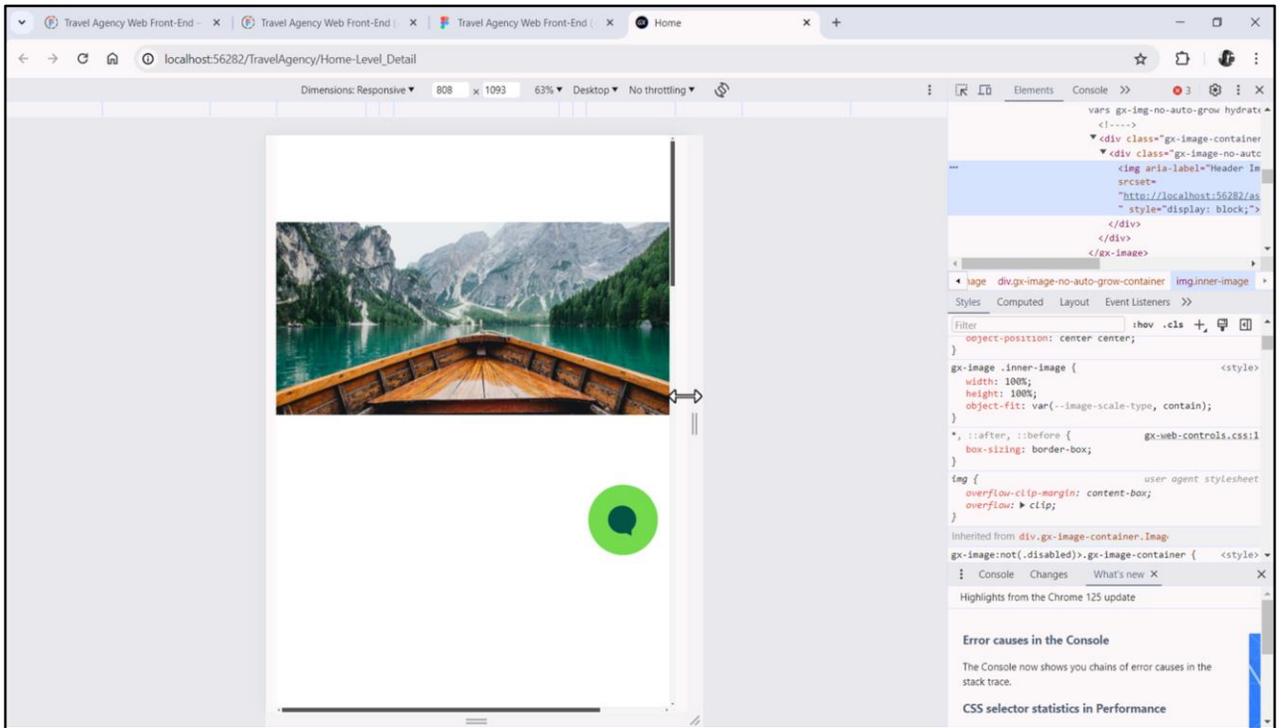
Highlights from the Chrome 125 update

Error causes in the Console

The Console now shows you chains of error causes in the stack trace.

CSS selector statistics in Performance

Ao contrário, se ampliarmos, permanece fixa, em seu tamanho exato.



E se diminuirmos, por outro lado, começa a diminuir proporcionalmente até entrar completamente no espaço que tem.

The screenshot shows a web browser window displaying a travel agency page. The page features a large image of a wooden boat on a turquoise lake, surrounded by mountains. Below the image, there is a green circular icon and text that reads "The new age of EXPLORATION". The browser's developer console is open on the right side, showing the HTML structure of the page. The console highlights the following code:

```
vars gx-img-no-auto-grow hydrate<
<!-->
  <div class="gx-image-container">
    <div class="gx-image-no-autc">
      <img aria-label="Header Im
        srcset="
          "http://localhost:56282/as
            " style="display: block;"
      </div>
    </div>
  </gx-image>
</div>
</div>
```

The console also shows the computed styles for the selected element, including:

```
object-position: center center;
}
gx-image .inner-image {
width: 100%;
height: 100%;
object-fit: var(--image-scale-type, contain);
}
*, ::after, ::before {
box-sizing: border-box;
}
img {
overflow: clip-margin: content-box;
overflow: clip;
}
Inherited from div.gx-image-container.Img:
gx-image:not(.disabled).gx-image-container {
}
Console Changes What's new X
Highlights from the Chrome 125 update
Error causes in the Console
The Console now shows you chains of error causes in the
stack trace.
CSS selector statistics in Performance
```

O que está acontecendo tem a ver com esta propriedade CSS, `object-fit`, que está assumindo esse valor `contain`.

The screenshot shows a web browser window with the URL `localhost:56282/TravelAgency/Home-Level_Detail`. The page features a large image of a wooden boat on a turquoise lake with mountains in the background. Below the image, there is a section titled "The new age of EXPLORATION" and two paragraphs of text. The browser's developer tools are open on the right, showing the DOM tree and the Styles pane. The selected element is `img.gx-image-no-auto-grow-container > img.inner-image`. The Styles pane shows the following CSS rules:

```
object-position: center center;

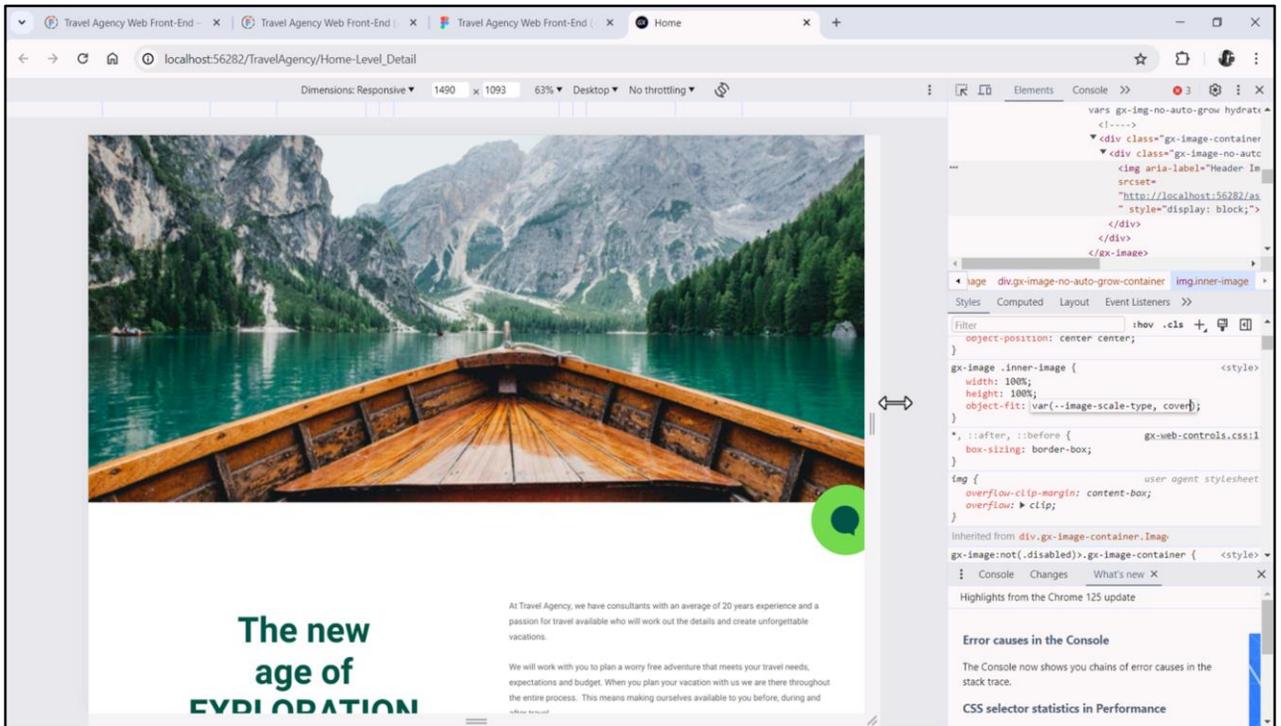
gx-image .inner-image {
  width: 100%;
  height: 100%;
  object-fit: var(--image-scale-type, contain);
}

::after, ::before {
  box-sizing: border-box;
}

img {
  overflow: clip;
  margin: content-box;
  overflow: clip;
}
```

The console shows "Error causes in the Console" and "CSS selector statistics in Performance".

Vamos ver o que acontece se a removermos... se estica ou comprime de modo a ocupar todo o container. Isto não é o que queremos.

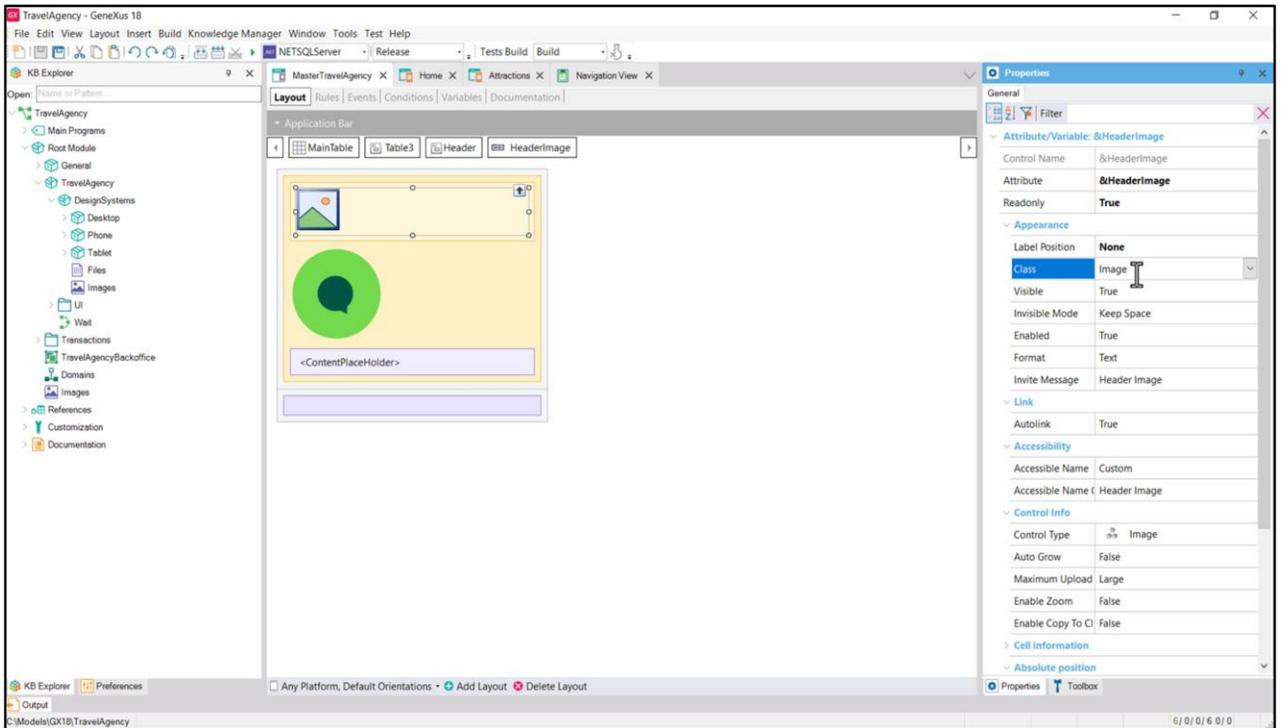


Queremos o comportamento de cover... Ou seja, que se estique ou comprima para ocupar todo o container, mas proporcionalmente, então terá que cortar alguma parte da imagem, obviamente.

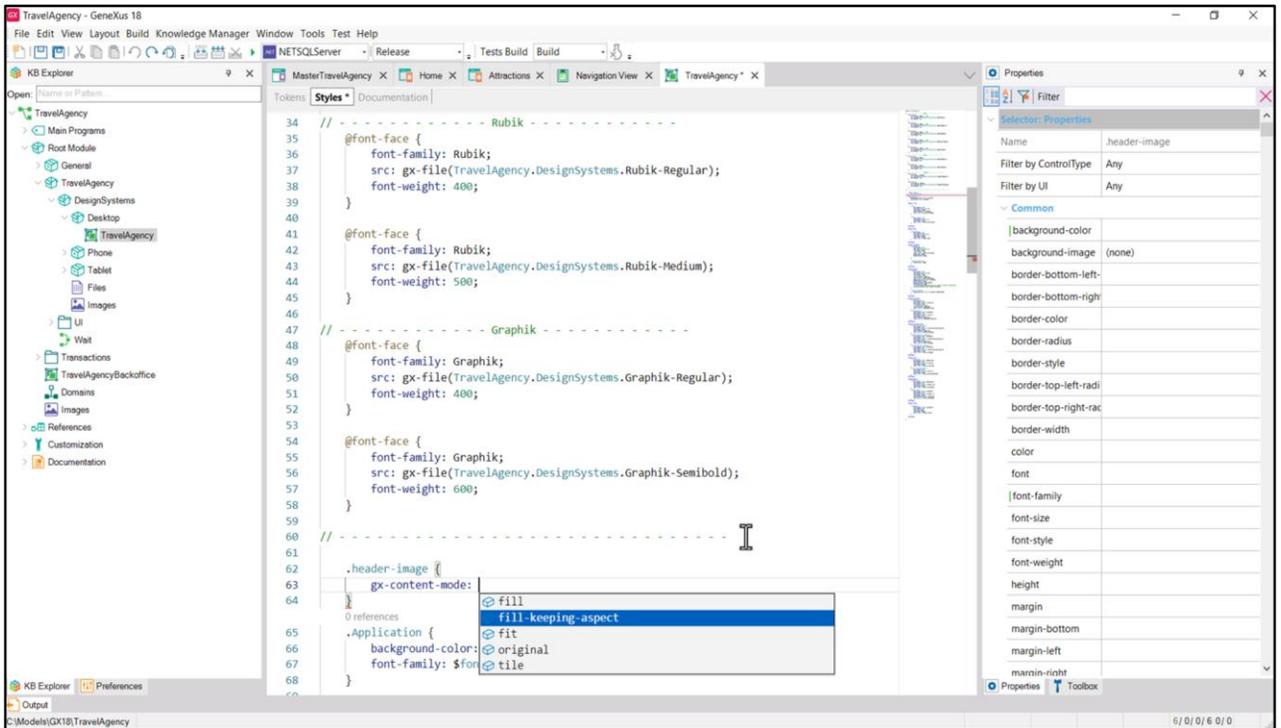
The screenshot shows a web browser window displaying the GeneXus Wiki page for the 'gx-content-mode property'. The page title is 'gx-content-mode property' and it includes a search bar, 'Sign up', and 'Login' buttons. The left sidebar contains a navigation menu with categories like 'Introduction', 'Look & Feel', and 'Images'. The main content area explains that the property specifies how an image is shown based on its size and the control area size. Below this, a table lists various values for the property.

Value	Description
	This is the default value. Indicates that no specific value is assigned to the property.
No Scale	Respects the original size of the image, independently of the control area size.
Responsive	Sets a responsive behavior.
Fill Keeping Aspect Ratio	The image makes bigger or smaller in width and height in order to fill the whole size of the control area, but keeping the aspect of the image. For example, if the image size is 100x200, and the control size is 50 x 50, then the image size is converted to 50 x 100.
Fill	The image is scaled in width and height in order to fill the whole size of the control area.

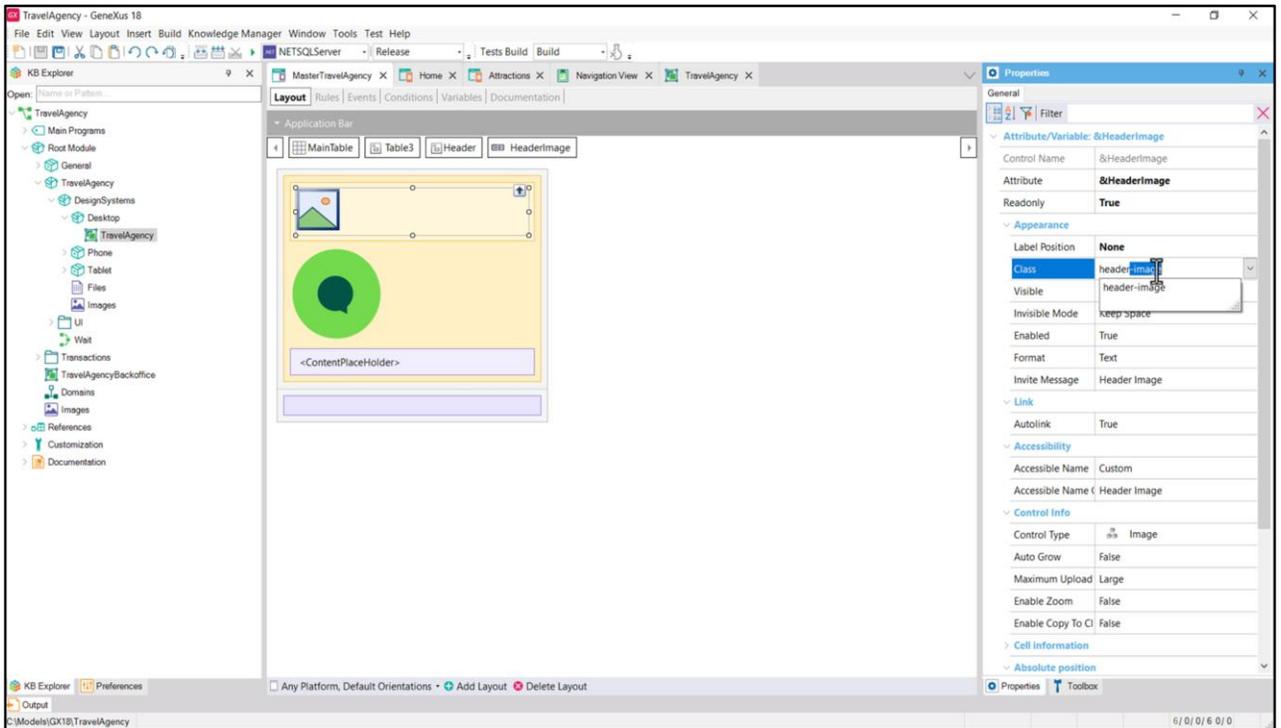
Esta propriedade CSS em GeneXus tem outro nome, para que seja transversal também a aplicações nativas. E é a **gx-content-mode**, que aqui vemos explicada na wiki de GeneXus, e vemos que se aplica tanto para Android, como para Apple e para Angular. Queremos esse comportamento: preencher, mas respeitando as proporções.



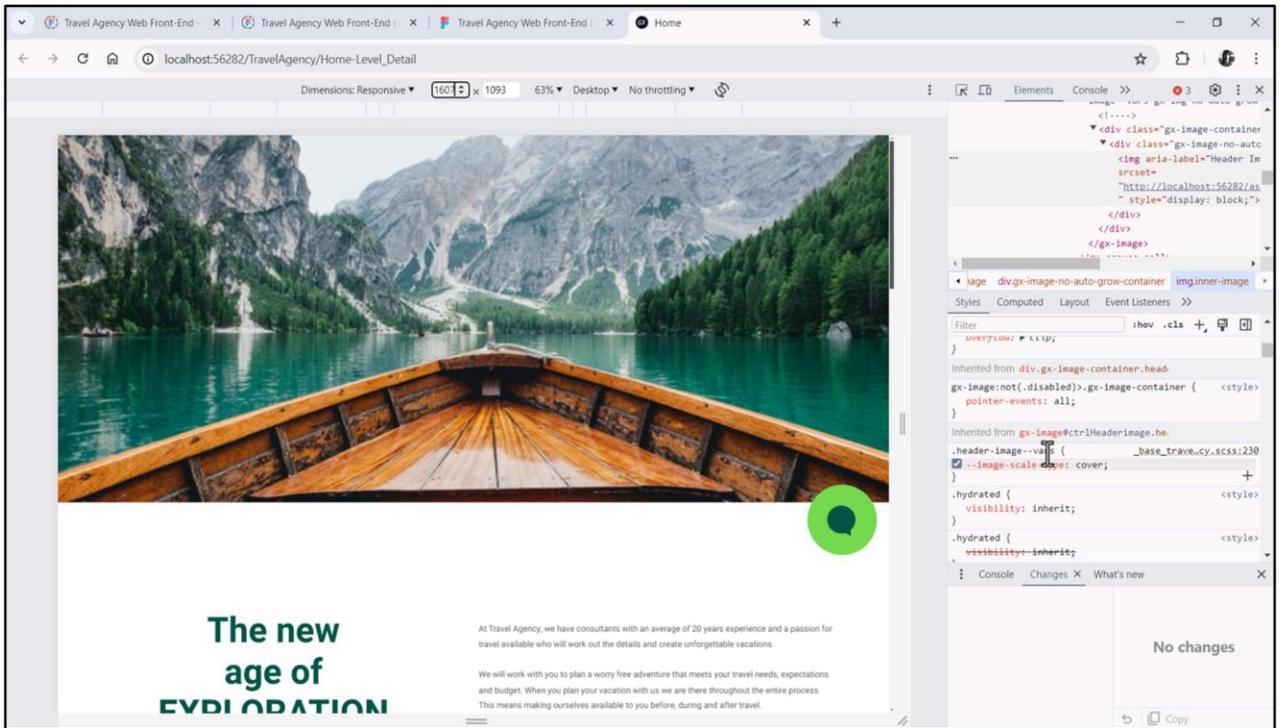
Portanto, precisaremos definir uma classe para a imagem, que contenha essa propriedade.



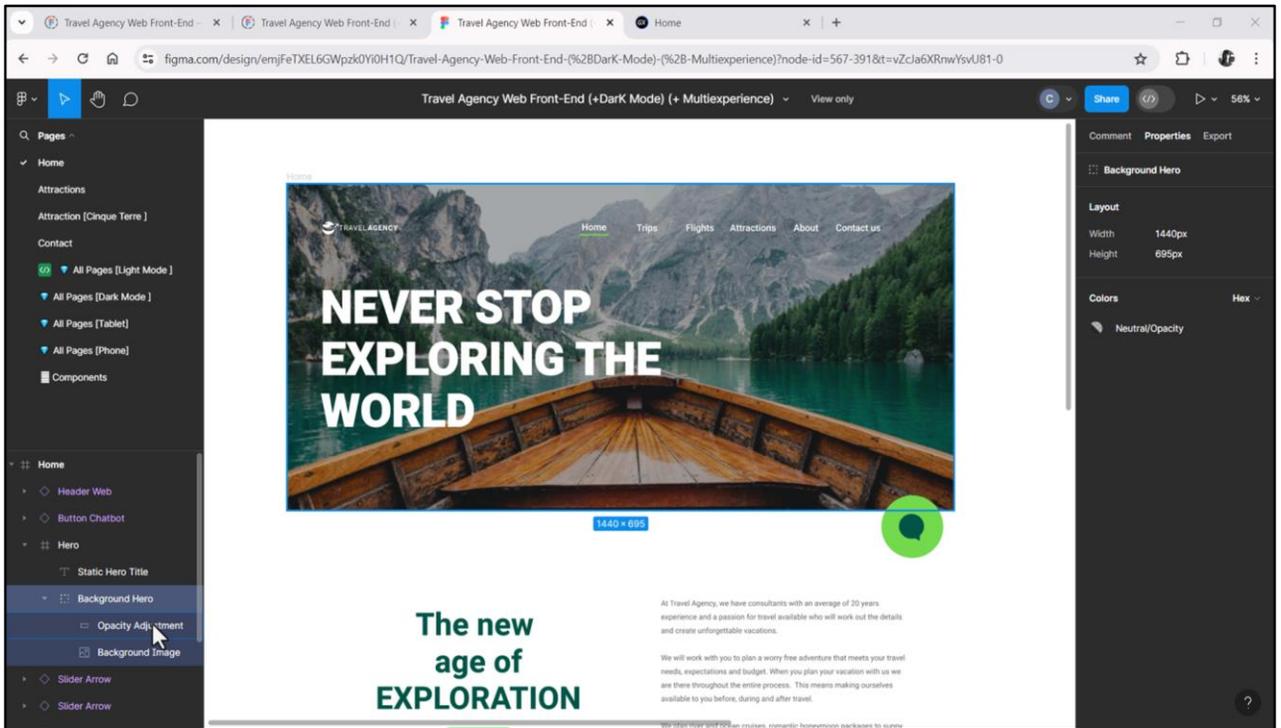
Então abro o DSO correspondente, é de Desktop, e em algum lugar específico a classe, que chamarei de header-image. E lá específico a propriedade e seu valor.



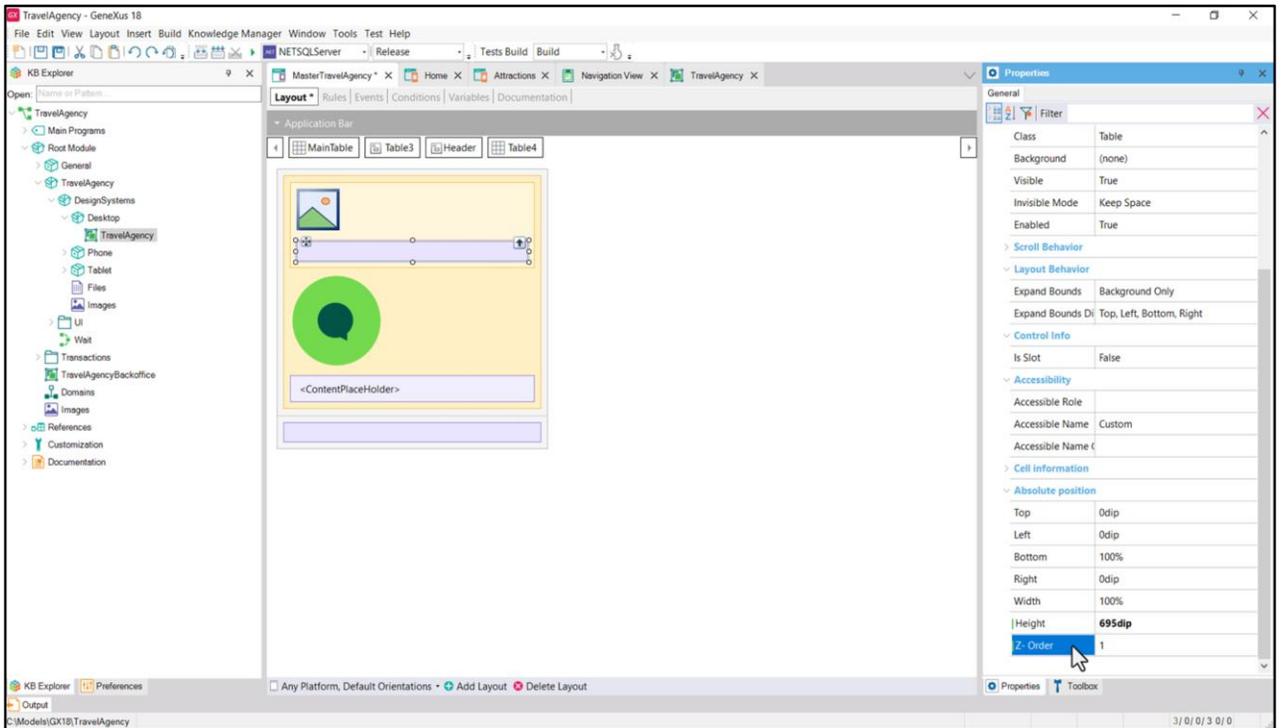
Em seguida, a associao ao controle no layout do Master Panel. Testamos?



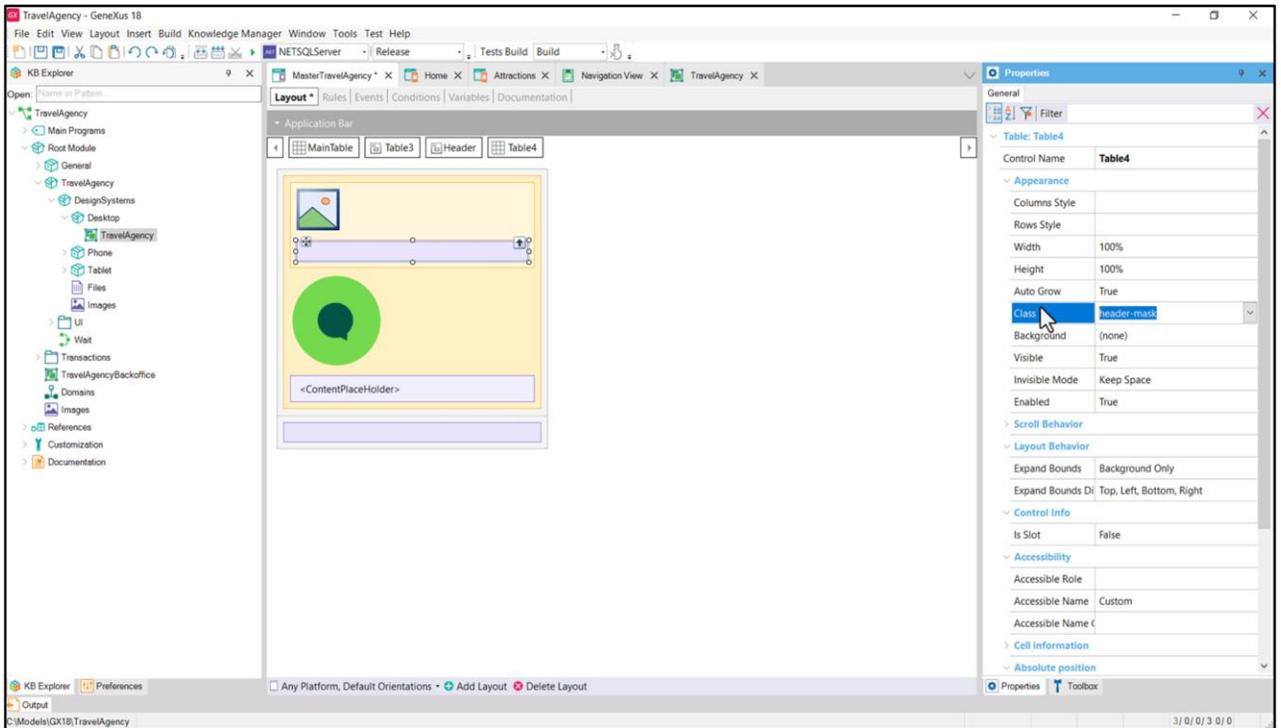
Agora sim, quando expandimos ou contraímos, vemos a imagem como queremos. Aqui podemos ver a classe operando, convertida pelo GeneXus para o código necessário. Não precisamos entender nada disso.



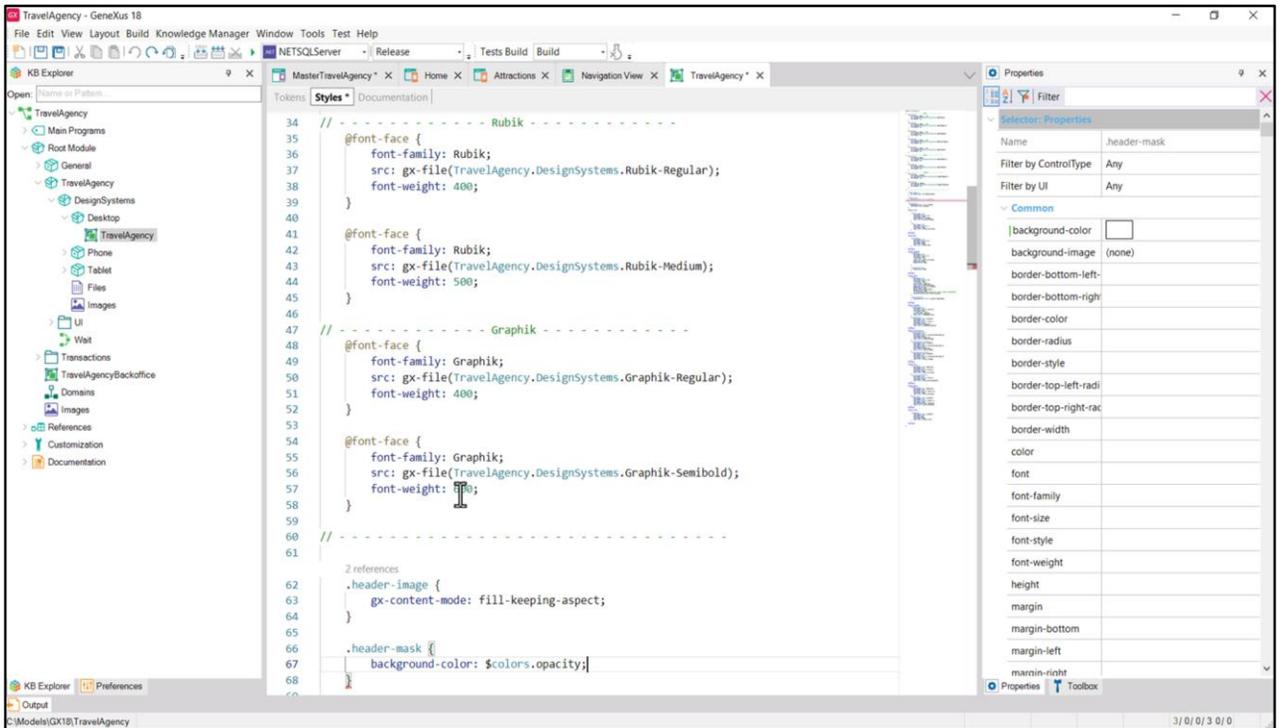
Outra coisa que já podemos perceber se formos ao Figma, é que há uma máscara sobre a imagem, para escurecê-la e conseguir um melhor contraste de todos esses elementos que vão ser sobrepostos a ela.



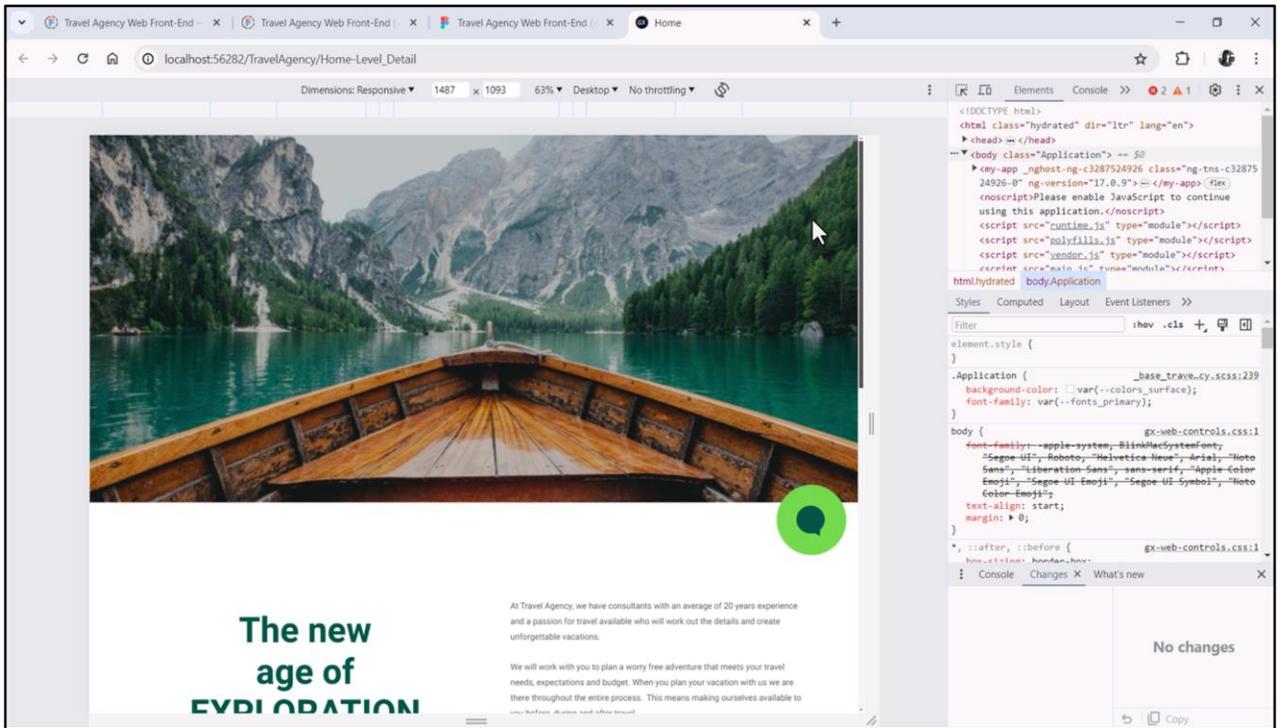
Uma maneira de implementar isso é através de uma tabela vazia, que a sobreponha, com exatamente as mesmas dimensões, ou seja, 695 dips de altura e o restante colado nas bordas do canvas, ou seja, 0 dips de cima, 0 da esquerda, 0 da direita e bottom 100%, que quando calculado quando for carregado também será 0, e podemos dar à Z-order o valor 1 para garantir que esteja sobre a imagem, que tinha 0 de posição z.



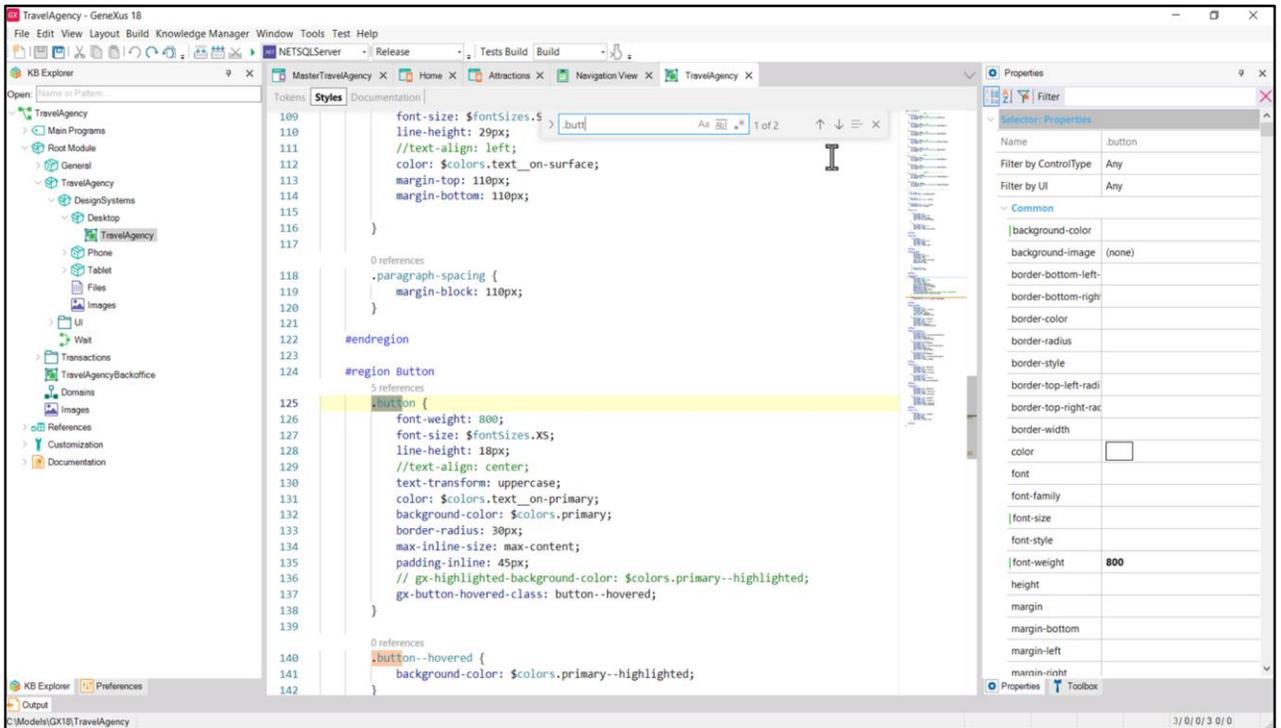
Em seguida, precisaremos especificar a background color da tabela por meio de uma classe. Então vamos chamá-la de header-mask.



E a especificamos no DSO, lembrando que já tínhamos o token de cor opacity definido. Em seguida, à propriedade background-color associamos esse token de cor.



Se executarmos agora, veremos a imagem com a máscara.

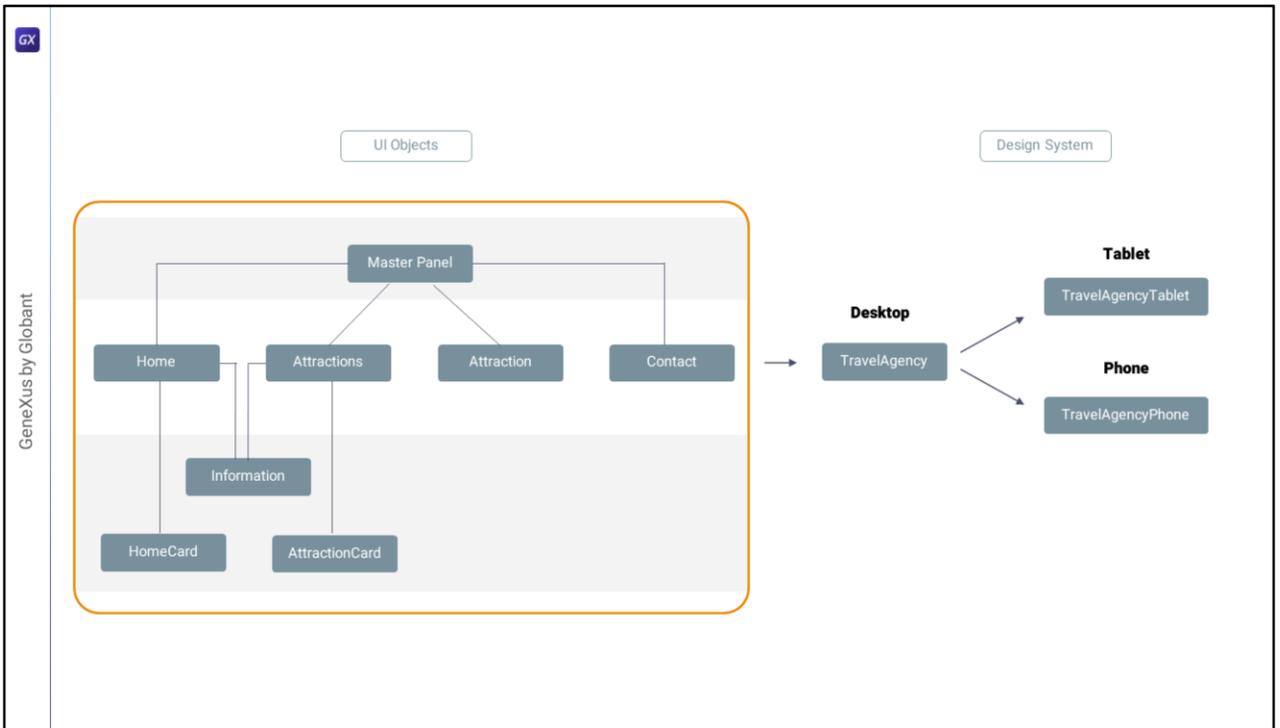


Antes de continuar, não sei se perceberam que está se tornando mais complexo o DSO. Quando fui especificar as duas classes que precisamos para criar o Header: a da imagem e a que usaríamos para a máscara, as escrevi depois das regras font-face, mas poderia tê-las escrito em qualquer lugar.

Se revisarmos o que tínhamos especificado neste DSO eram, além das regras para incorporar as fontes, todas as classes para os estilos tipográficos de toda a aplicação, e também para a do botão de contato, aquelas que controlavam o estilo do botão.

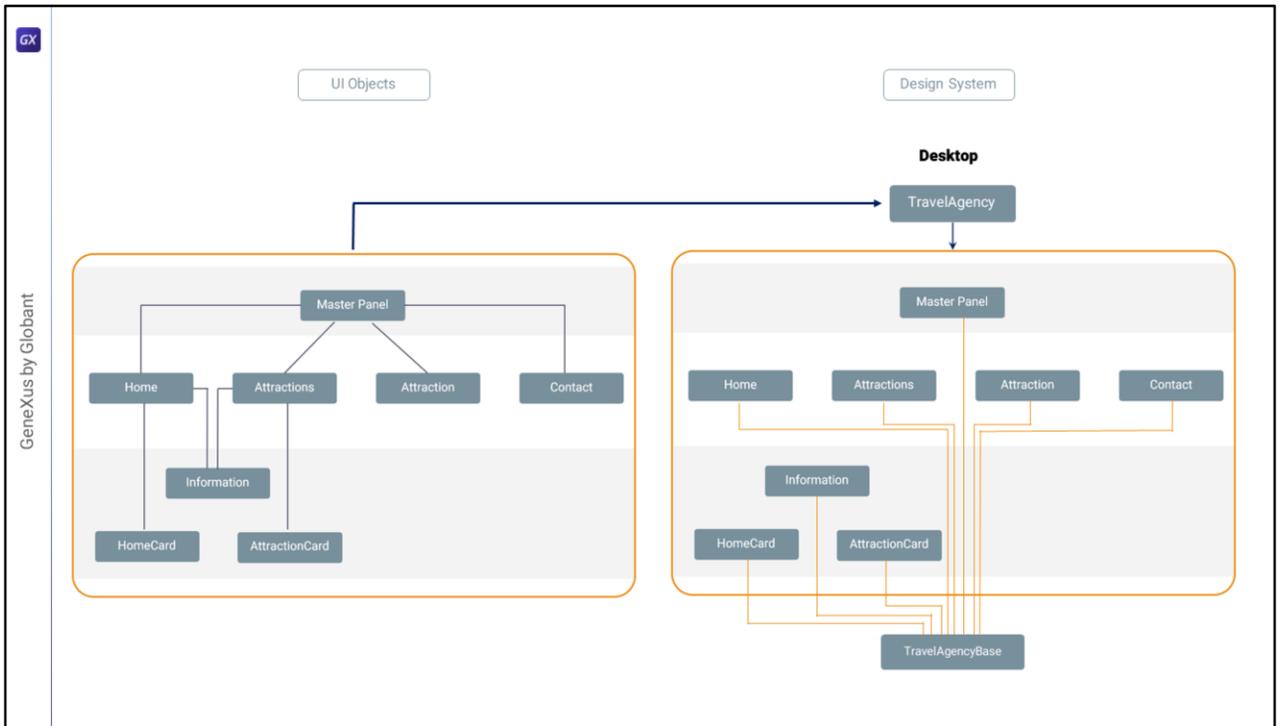
Mas agora precisamos começar a incorporar classes para os distintos controles de todos os layouts. Imaginem que isso vai crescer bastante. O editor nos dá a possibilidade de fazer pesquisas com control F.

Mas para tornar tudo isso mais administrável e compreensível, uma possibilidade que é aquela que vou escolher (depois falarei sobre isso um pouco) será também de alguma forma componentizar o DSO.



Deixe-me explicar: tudo o que fizemos até agora foi especializar o DSO TravelAgency para o que será o tamanho Tablet, por um lado, e por outro o tamanho Phone.

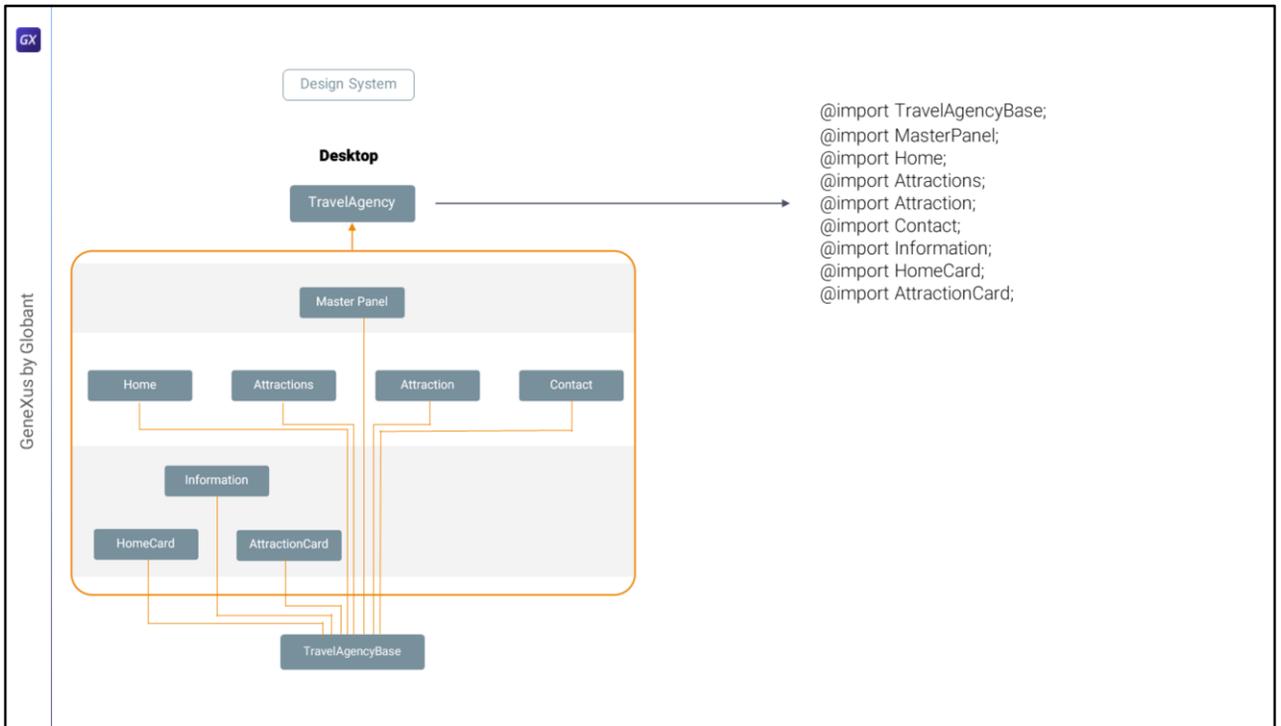
Mas na realidade nada nos impede de ter uma árvore de DSOs para cada um e não um único objeto.



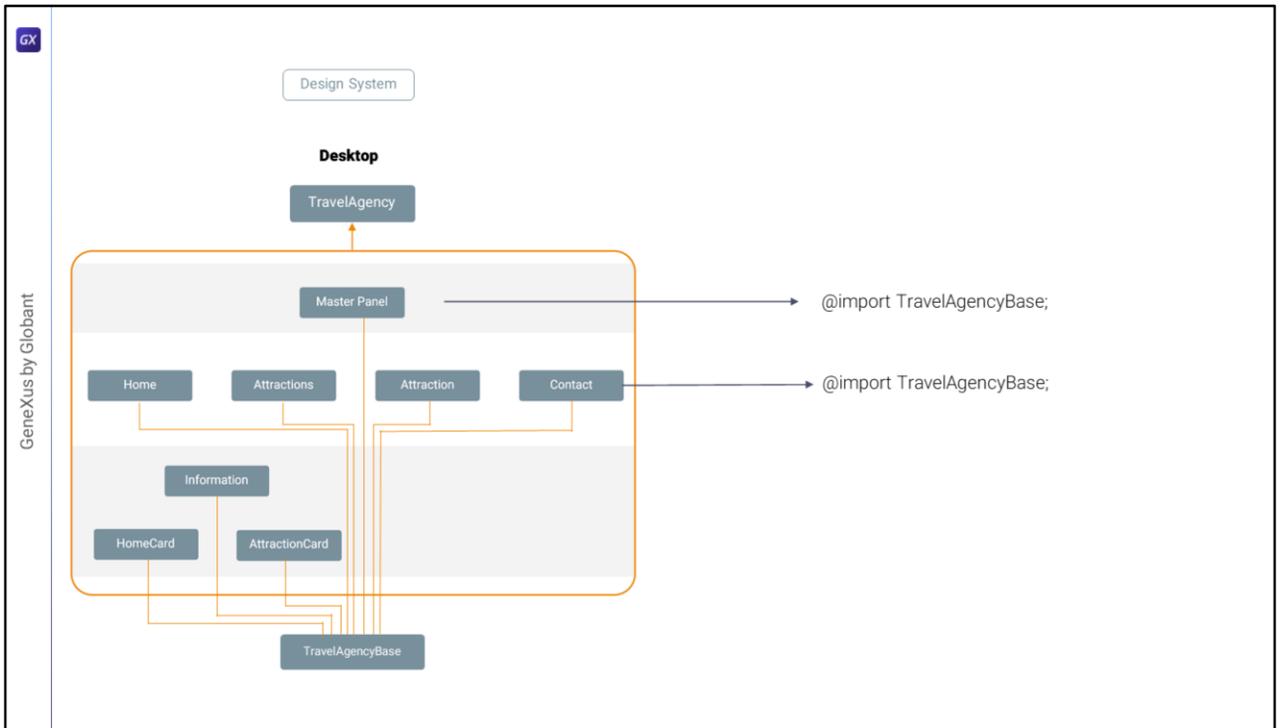
Ou seja, vamos pensar apenas no tamanho Desktop... Não seria muito mais organizado, por exemplo, ter um DSO base, com todas as definições transversais a toda a aplicação, como, por exemplo, a inclusão das fontes, e toda a análise de tokens que já fizemos, e até, se quiser, também poderíamos deixar lá todas as classes para a tipografia... que é basicamente o que fizemos até agora no DSO TravelAgency... bem, que esse seja um DSO base e então ter tantos DSOs quanto objetos, que definam, cada um, as classes que aplicarão aos controles de seu layout, de seu próprio layout.

Cada um importando as definições gerais do DSO Base, e especializando o que precise e definindo, claro, suas classes próprias, as que são próprias do layout desse objeto. O critério deveria ser: tudo o que seja comum a vários objetos deveria ir no DSO Base e o que é absolutamente específico, no DSO do objeto.

Então, como não podemos dizer a cada objeto da UI qual é seu DSO, mas sim que é um único DSO para toda a plataforma, será suficiente ter o DSO pai, Travel Agency, que a única coisa que fará será importar todos esses outros.



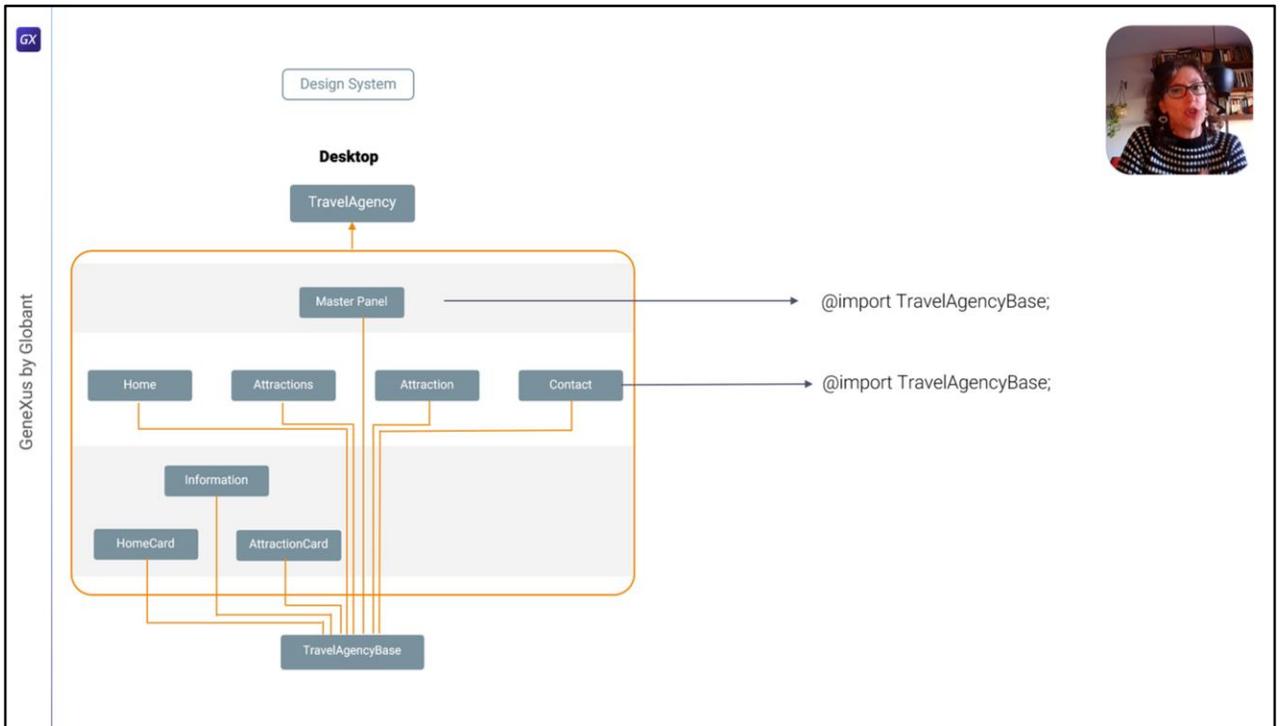
Resumindo, o DSO TravelAgency terá apenas tantas regras import quanto DSOs para cada objeto tenhamos definido, mais um import do DSO Base (o que em princípio não seria necessário porque será importado pelos outros, mas conceitualmente está bom).



Em seguida, o DSO que dá estilo particular a cada objeto provavelmente precisará importar o DSO base, que tem as definições gerais. Assim o do Master Panel, o de Contact e todos os outros.

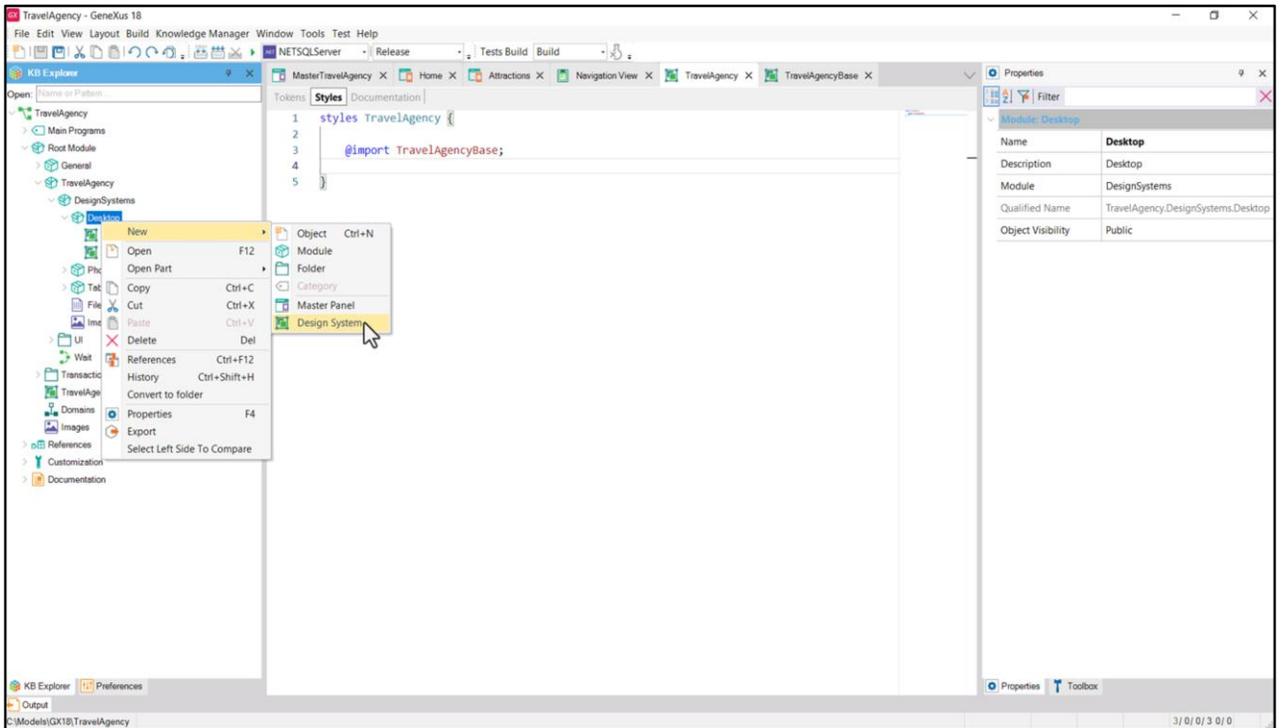
Poderíamos pensar que importar o DSO Base aqui, e também em cada um dos DSOs, irá repeti-lo desnecessariamente e então penalizará a aplicação gerada, no caso de Angular, com um CSS enorme. Não será assim. No CSS final, aparecerá apenas uma vez o DSO Base.

Teremos que ver até qual granularidade chegamos (estou pensando nos stencils, por exemplo). Aqui sempre se trata de soluções de comprometimento.

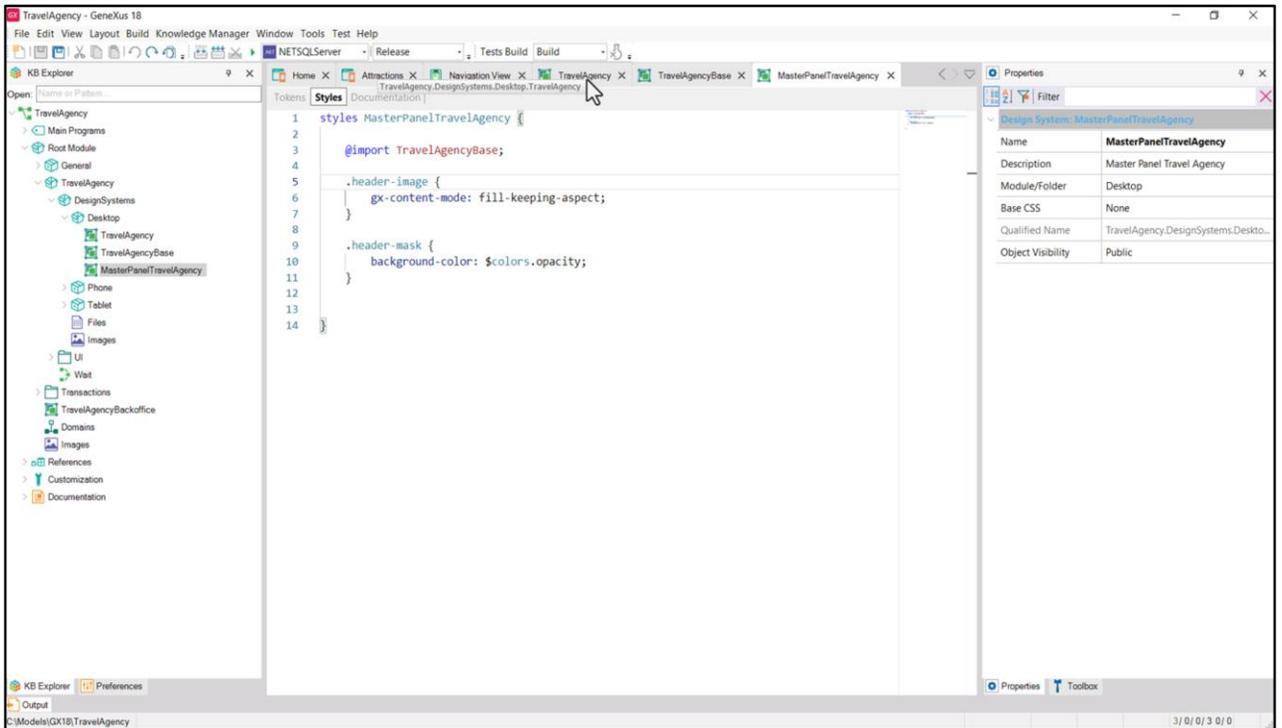


Esta que estou propondo é uma possível solução para enfrentar a complexidade, que não tem consensos. De fato, eu a submeti para discussão com vários colegas que estão trabalhando permanentemente no frontend dentro da equipe de GeneXus e não houve muito consenso.

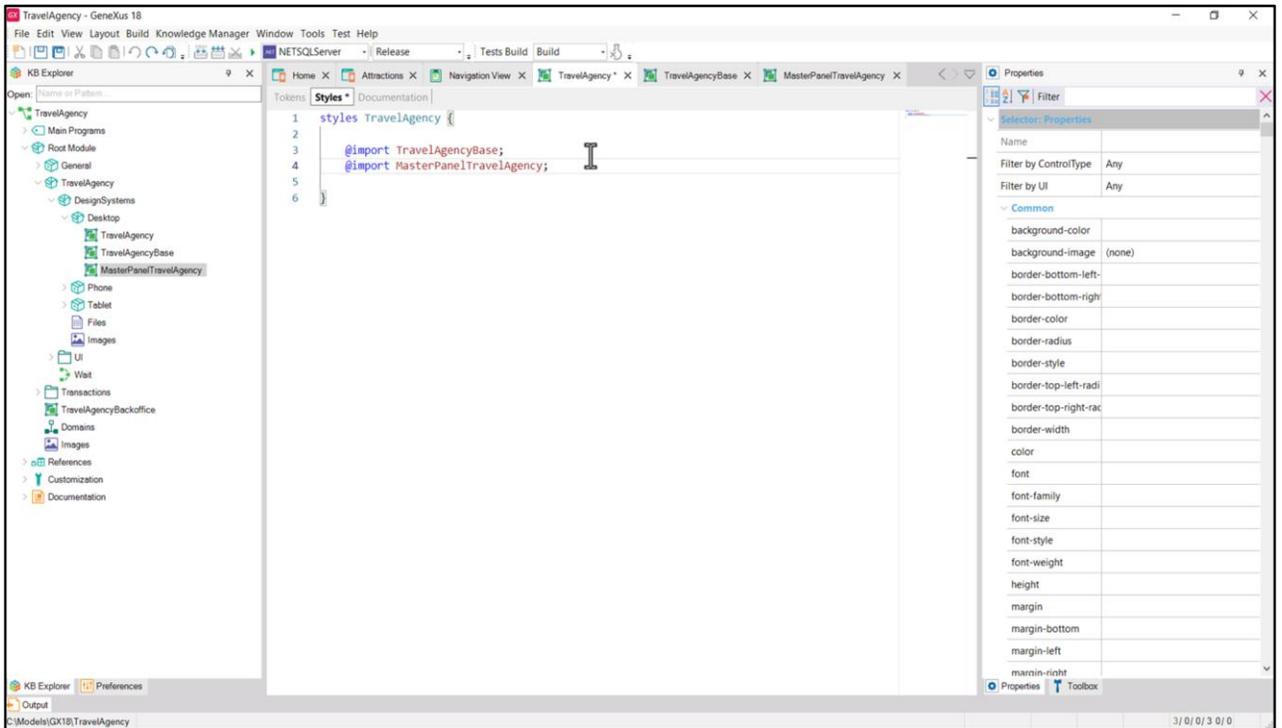
Mas a deixo em aberto para discussão, este não é o momento de resolvê-la, obviamente, mas queria apresentá-la a vocês porque é a que vou começar a utilizar e, em todo caso, mais tarde poderemos questioná-la.



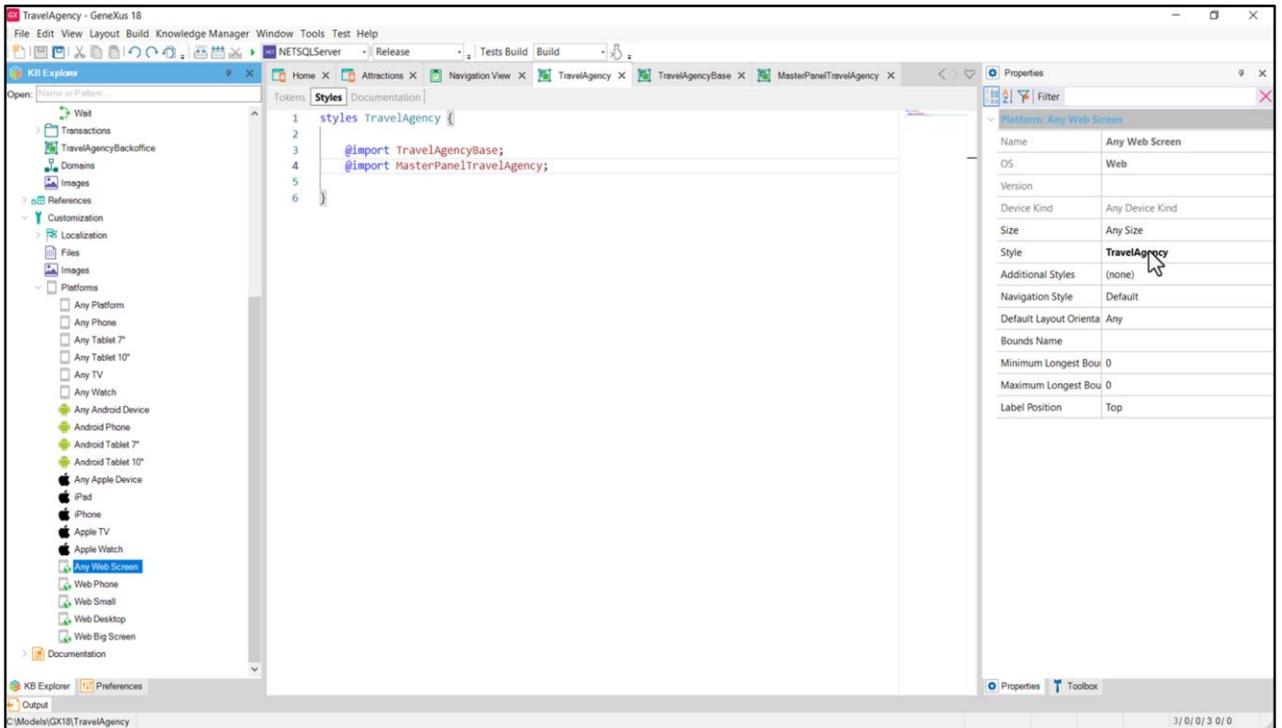
Em resumo, então, vou salvar esse objeto com o nome TravelAgencyBase.
Vou esvaziar o TravelAgency e o que vou fazer é, no momento, importar este outro.
E por outro lado vou criar outro DSO...



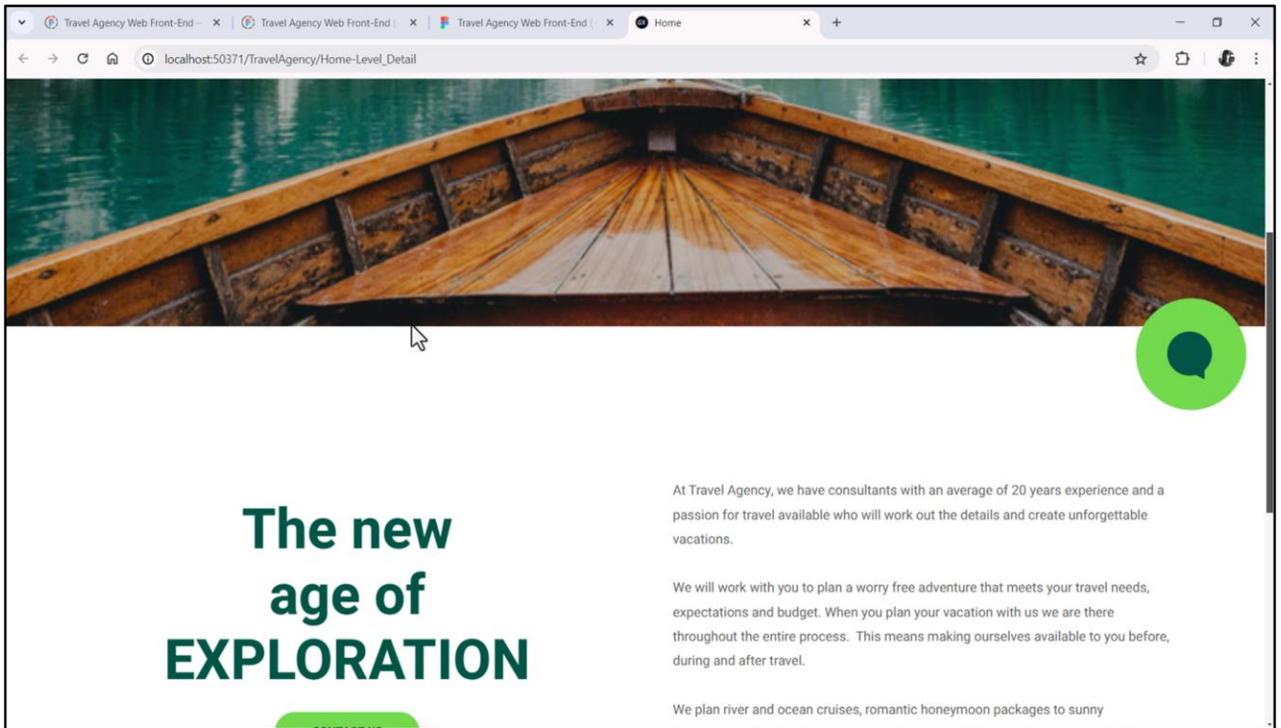
...que chamarei da mesma forma que o Master Panel. Nele importarei o TavelAgencyBase, e moverei deste, as classes que especificamos para o Header no momento.



E, por último, importo também no DSO pai, este outro.

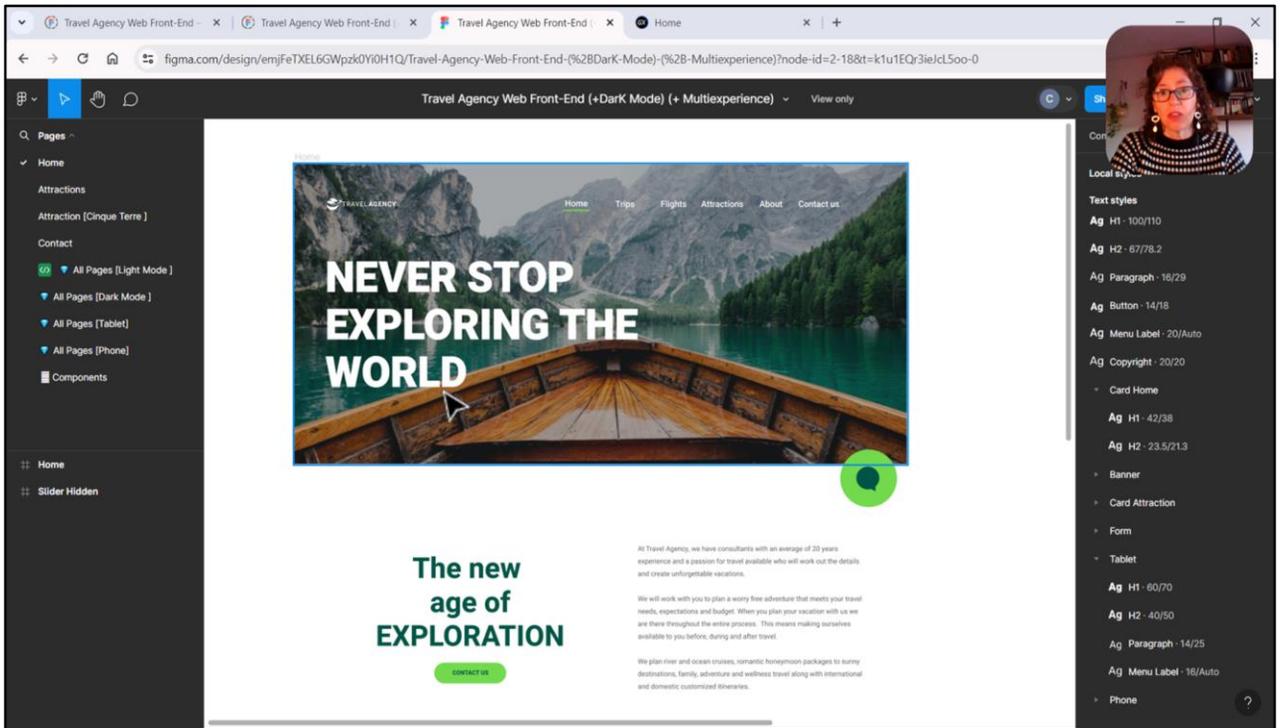


Tudo deveria continuar funcionando como está, já que a plataforma em que estamos executando continua tendo como seu DSO este, TravelAgency, que agora importa estes outros dois. Então ele conterà tanto os tokens e classes do Base, quanto as deste outro.



Vamos testar executar para verificar se tudo parece exatamente igual.

E sim, tudo parece igual.



Bom, como esse vídeo já ficou um pouco longo, vamos continuar no próximo com o texto que queremos sobrepor na imagem, o logotipo e o menu.

GX

GeneXus by Globant

GeneXus[™]
by Globant

training.genexus.com