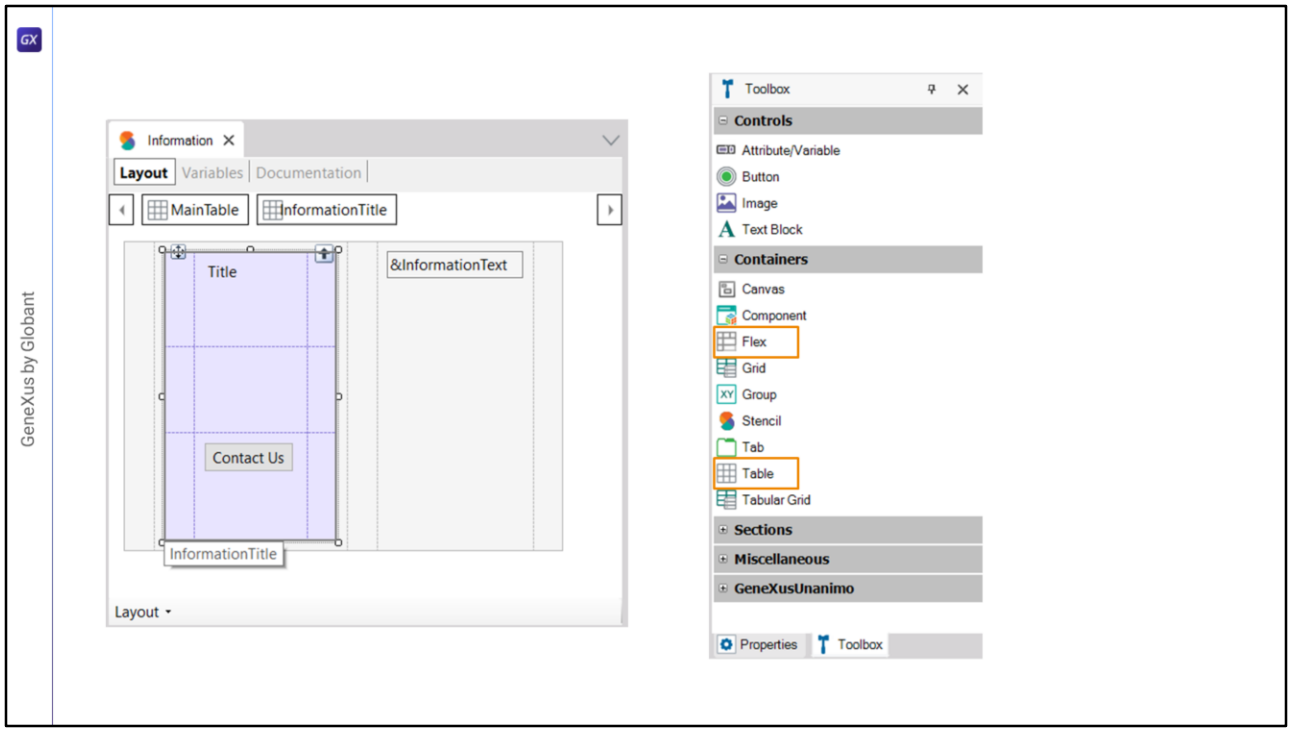
GeneXus by Globant

First Layout in GeneXus

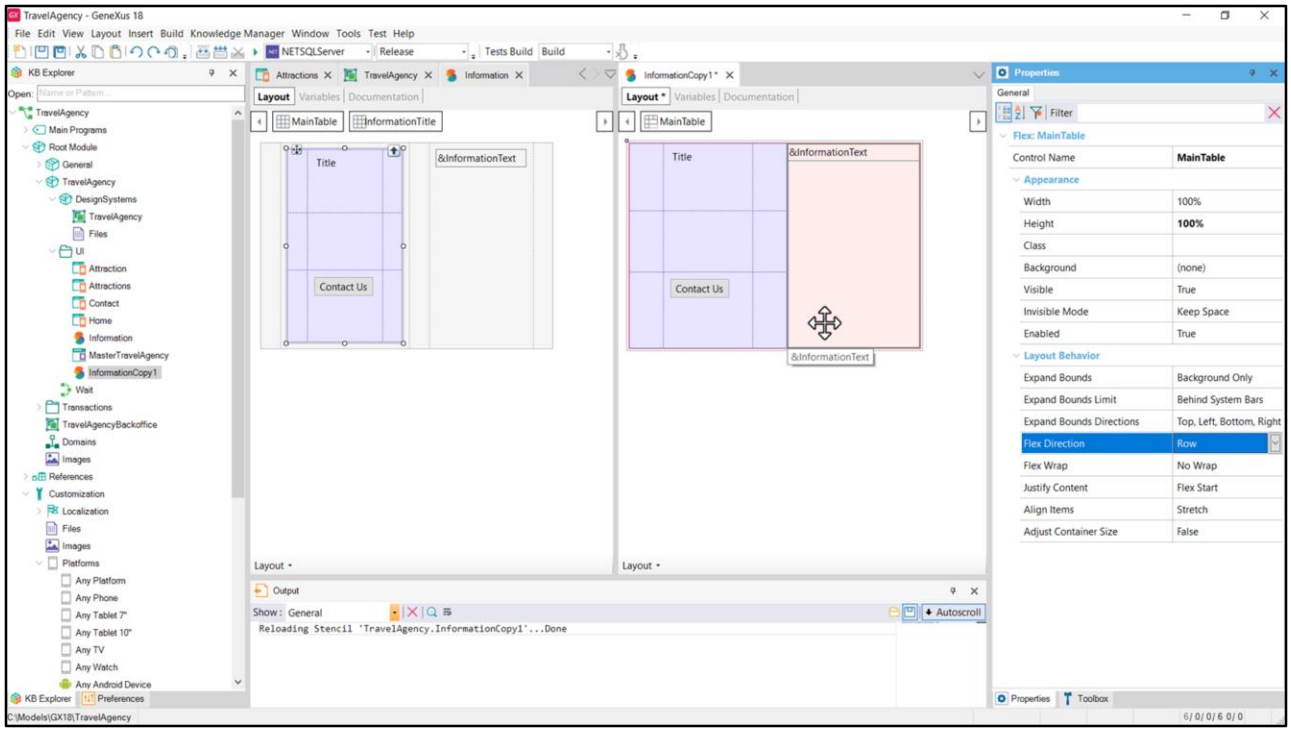
Flex control instead of Table

Cecilia Fernández



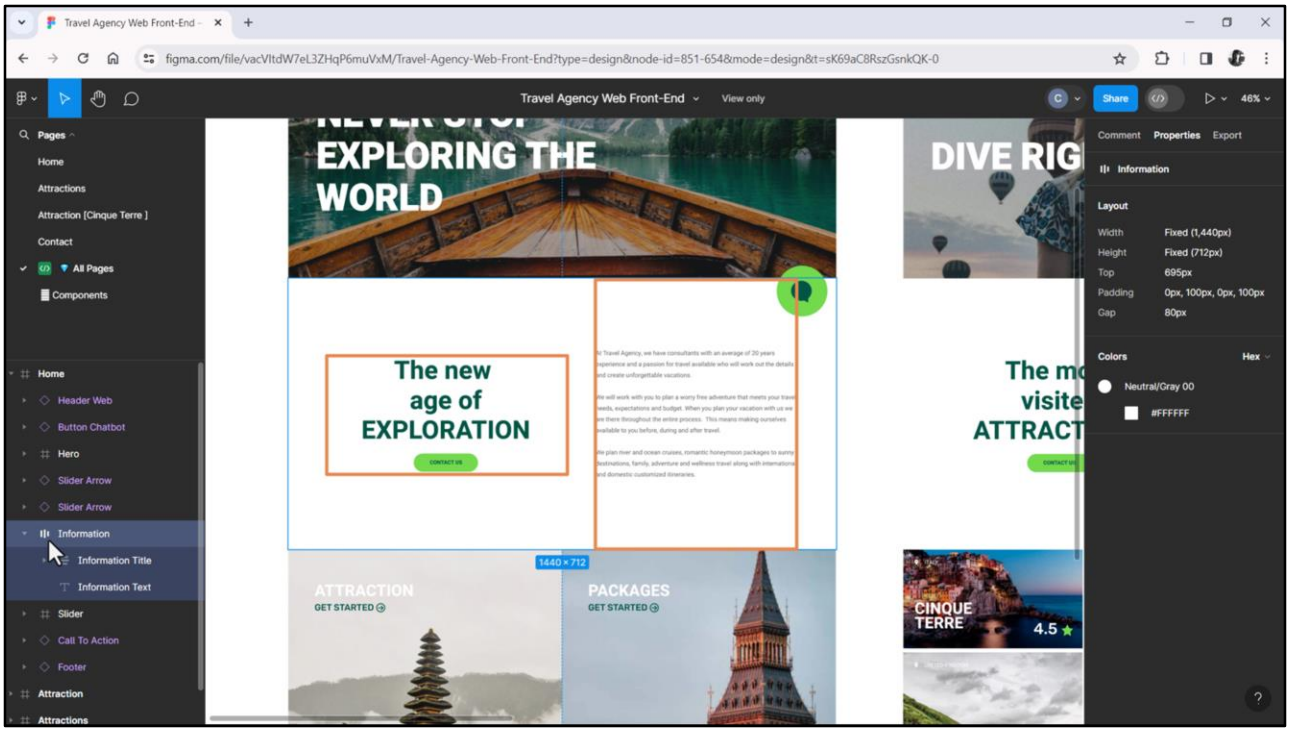
Quando construímos o Stencil, no início deste módulo, optamos por modelá-lo a partir da tabela principal e de outra tabela para o textblock e o botão.

Mas ali mesmo falamos que também poderíamos ter escolhido como container, não o controle Table e sim o controle Flex, lembram?



E mais (vou fazer um save as de nosso stencil com outro nome)... havíamos dito que poderíamos converter a qualquer momento uma tabela comum como essa em Flex e vice-versa. Ao convertê-la vemos que desaparecem as linhas e colunas, e aparecem estas propriedades dentro do grupo Layout Behavior. A que a princípio, chama primeiro nossa atenção é a Flex Direction, cujo valor default é Row, e é o que vai determinar que os dois elementos contidos pela flex se disponham horizontalmente ao longo de uma única linha, **sem** colunas.

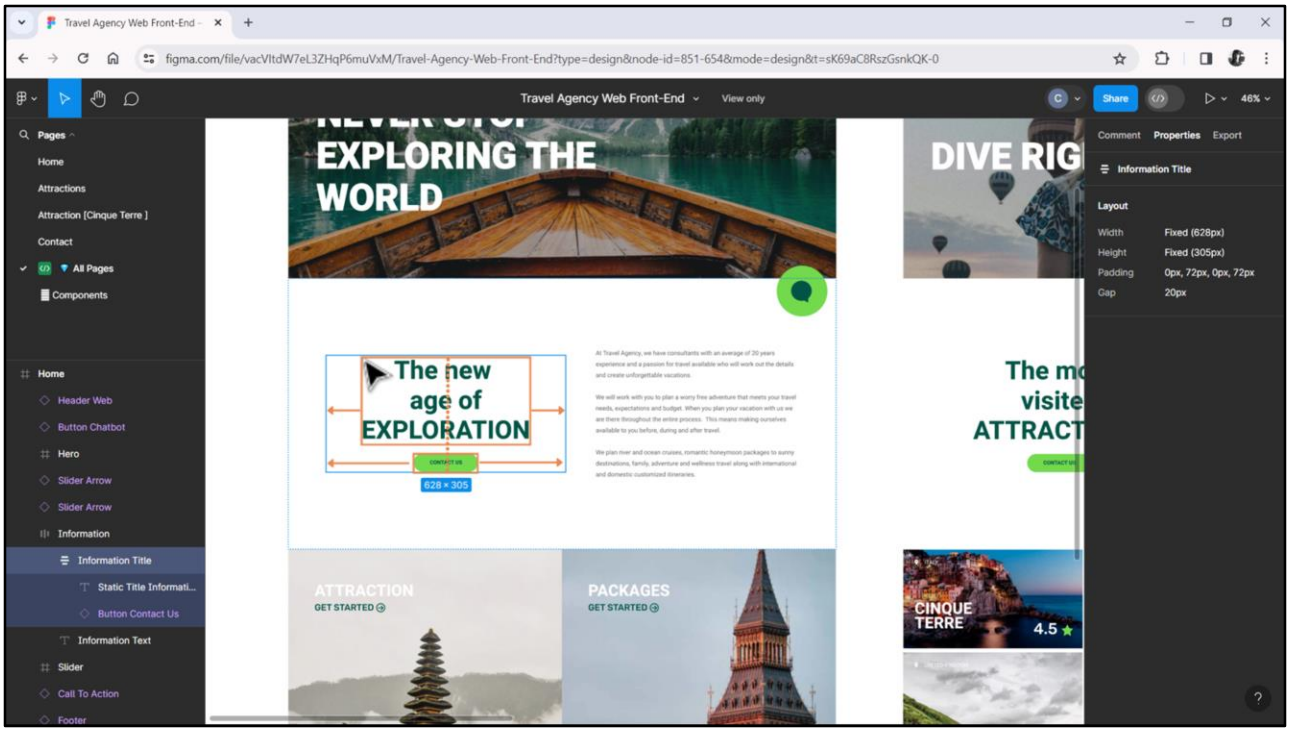
Ficará um pouco mais claro em Figma.



Para o frame information aparece este símbolo e para o frame que contém o título e o botão este outro símbolo. Estes símbolos nos dizem muito mais do que pode parecer.

O primeiro está nos indicando que os elementos do frame (este e este) se colocarão ao longo de uma linha, ou seja, horizontalmente: primeiro um e depois o outro. E também, que esses elementos podem ter diferentes alturas, mas serão alinhados pelo centro.

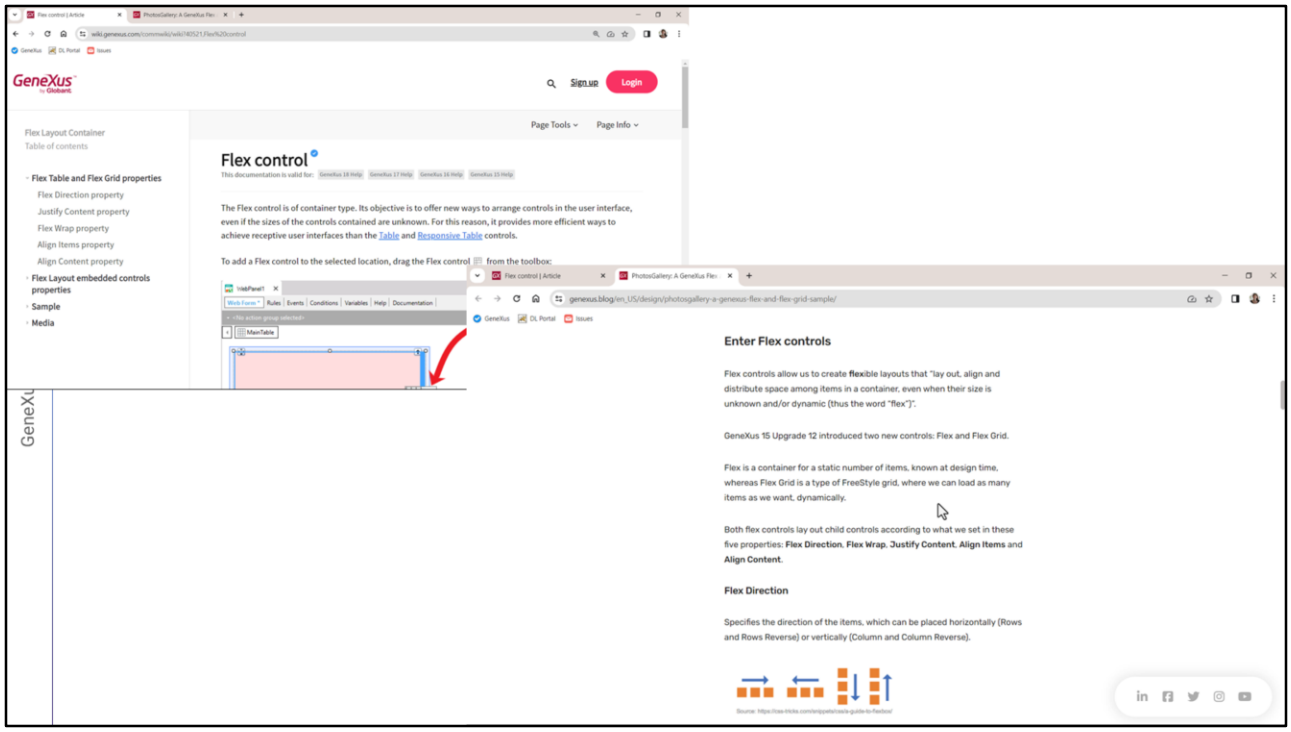
Na verdade, têm alturas diferentes: se virmos este... tem esta altura e se virmos este outro tem esta outra altura, que é a do container. Mas ambos estão alinhados verticalmente em relação ao centro.



Se observarmos este outro, vemos que também se trata de um container que contém dois elementos, desta vez dispostos em torno da direção coluna. Primeiro um, o texto, depois o outro, o botão. E que também estão alinhados pelo centro em relação às bordas da outra direção. Assim como estes dois, cuja direção é row, estão centralizados em relação à vertical.

Chechu consegue isto dizendo que o container tem, o que em Figma é conhecido como Auto Layout. Para nós isso se chama: Flex.

Basicamente significa que os elementos contidos serão colocados ao longo de uma direção: quer seja horizontal, como esta, ou vertical, como esta outra.

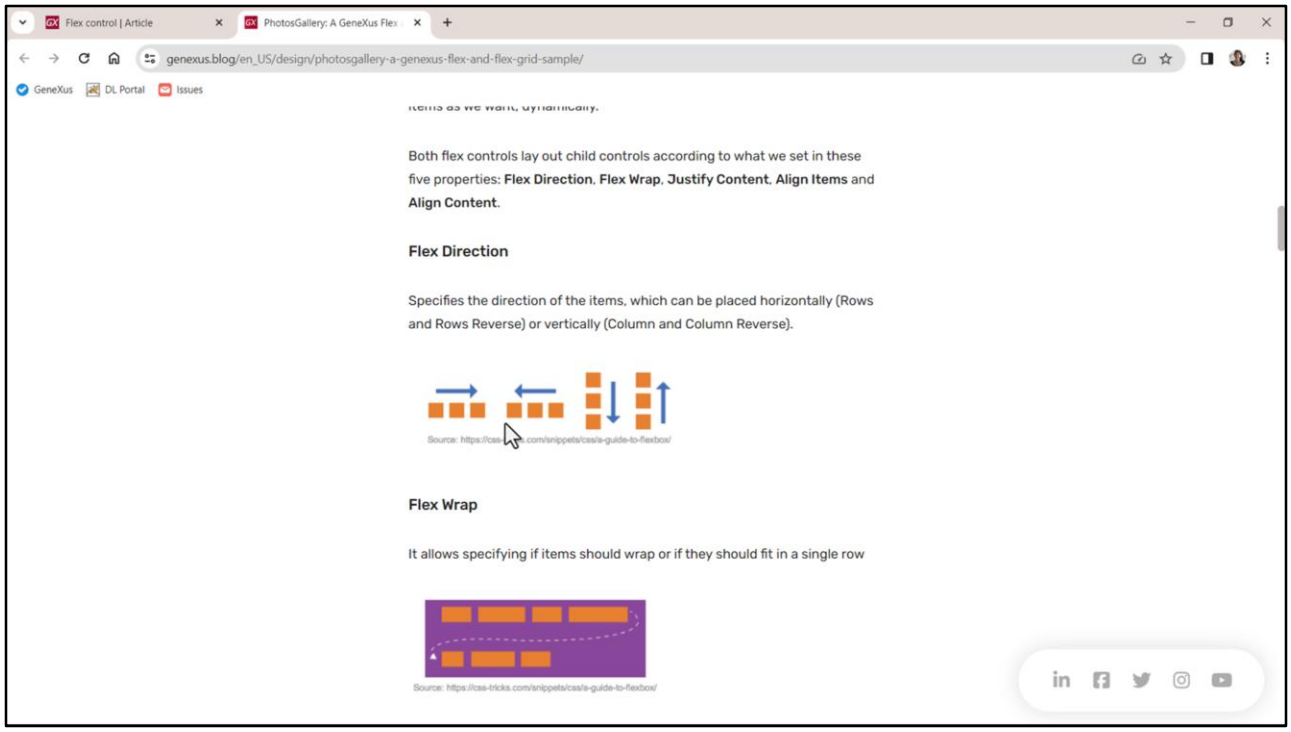


Nesta página da wiki de GeneXus se explica o controle. E é oferecido ao final um link para esta outra página que o explica com um exemplo.

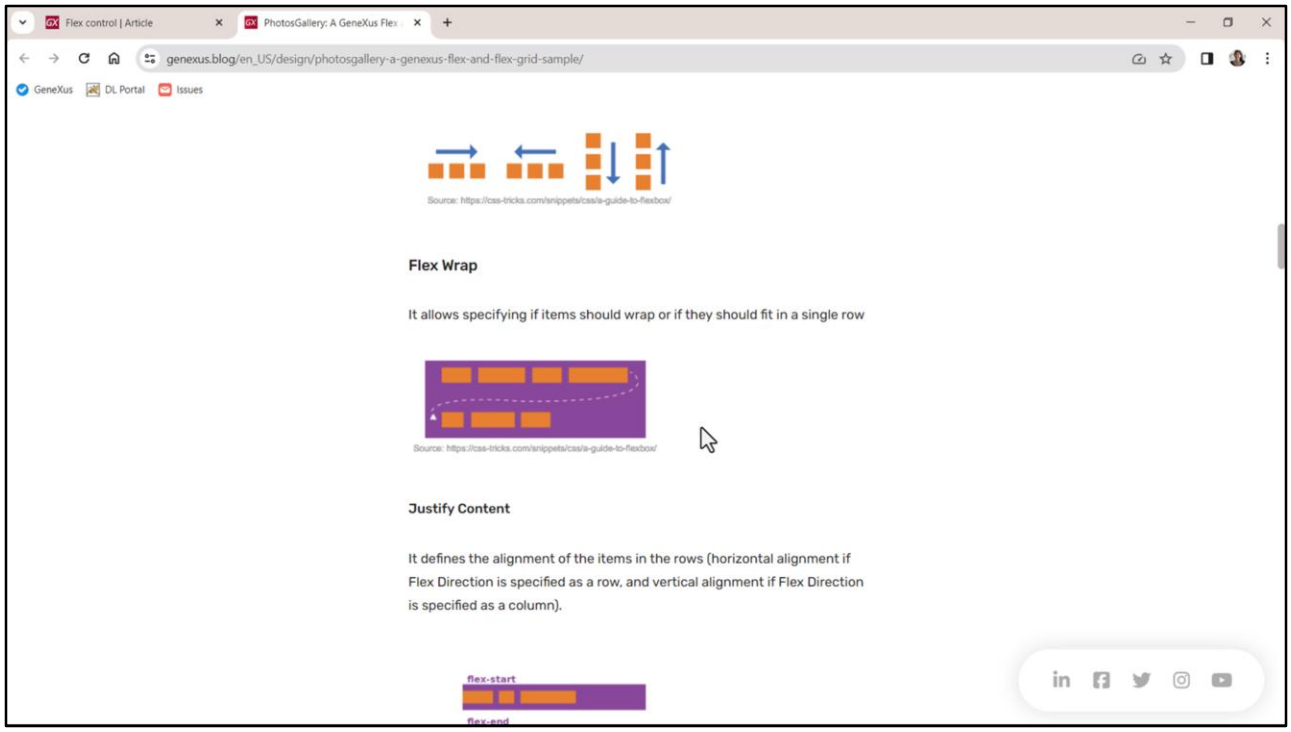
Vou aproveitá-la para que vejamos graficamente a questão, assim nos vai ficar mais claro.

Aqui diz que estes containers permitem criar layouts **flexíveis**, que permitem colocar os itens, alinhá-los e distribuir o espaço entre eles em um container, claro, mesmo (ou sobretudo, eu diria) quando seu tamanho é desconhecido a priori ou é dinâmico.

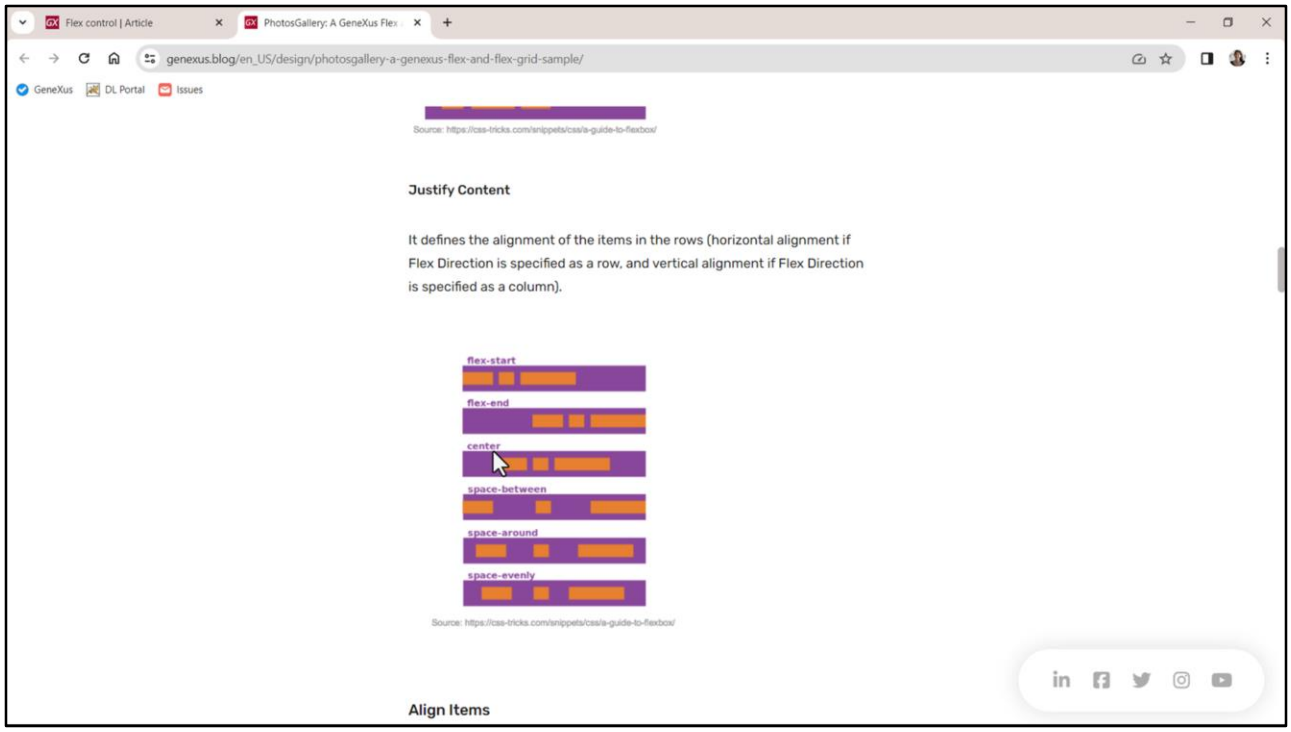
Aparece, então, uma série de propriedades para configurar direção e questões da distribuição dos controles.



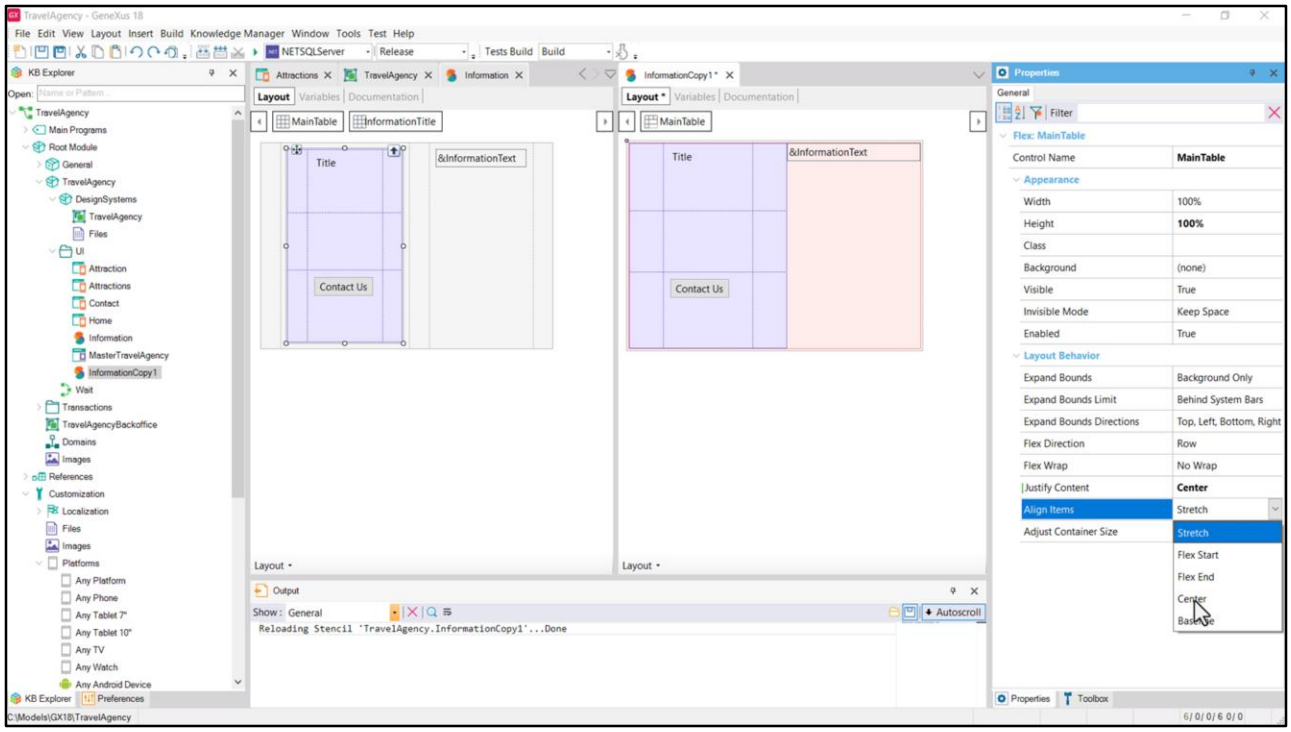
Por exemplo, este seria o caso de um container em direção row, este outro row reverse, este em direção column e este outro em column reverse. No exemplo, cada um dos containers contém 3 itens.



Neste outro exemplo dizemos se é permitido ou não que seja feito wrap, se o tamanho dos itens, ou seja, a largura que ocupam, somados, ultrapassa os limites do container. Então, neste caso estamos permitindo o wrap, ou seja, que se estenda em uma linha abaixo, visto que a direção é row, claro. Se a direção fosse coluna, se estenderia em uma segunda coluna.



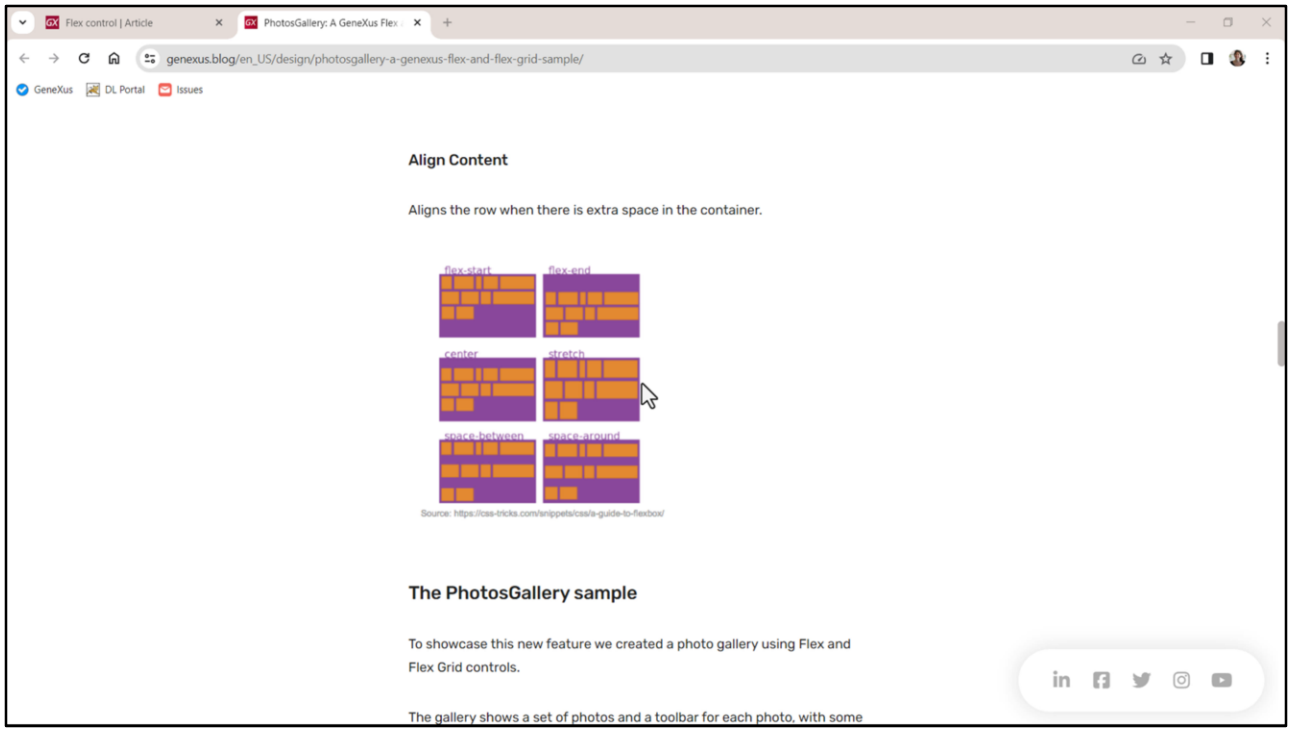
Esta outra propriedade indica como você deseja que o conteúdo seja justificado. Por exemplo, se quisermos que os itens sejam centralizados em relação à direção, que neste caso é row. Esse justamente seria o caso de nosso flex Information.



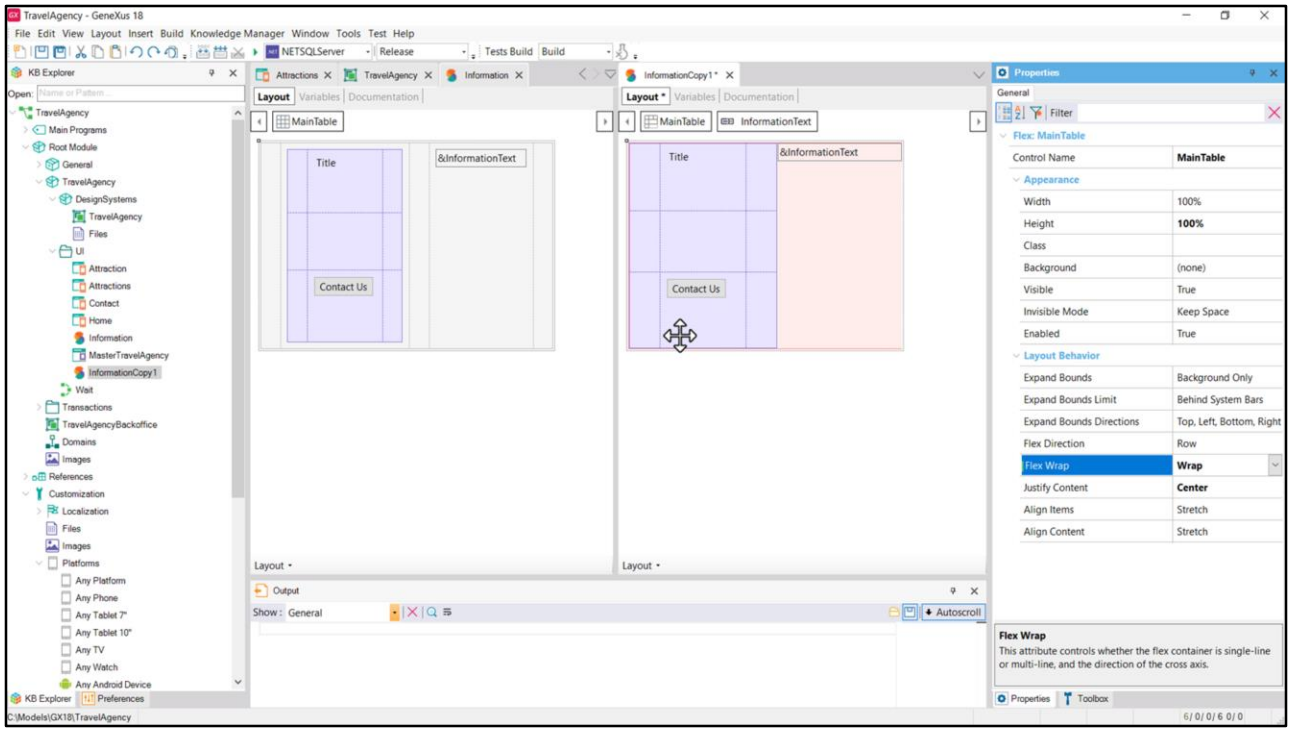
Então viemos aqui, vamos fazendo... Flex Direction “Row”. Flex Wrap vamos deixar um “No Wrap” a princípio. E a justificação “Center” (vemos as opções).

E o alinhamento dos itens em relação a acima e abaixo diremos que também seja centralizado.

E aqui ficará claro para nós o que estamos dizendo com isso.



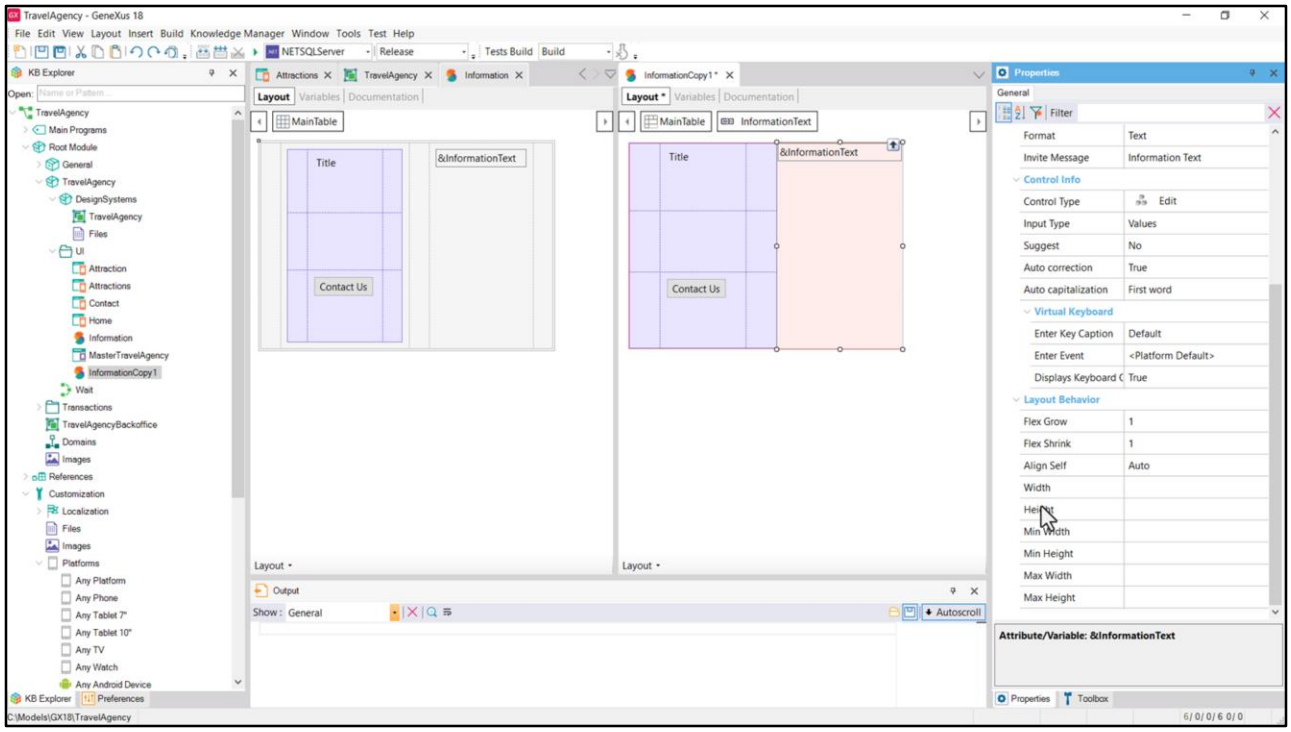
Por outro lado, se permitirmos o wrap, com esta propriedade determinamos como ajustar o conteúdo em relação ao espaço restante.



Bom, mas também não pretendia contar todas as possibilidades do controle, mas simplesmente apresentá-lo porque é um controle muito utilizado tanto em web quanto em aplicações nativas, justamente pela sua flexibilidade.

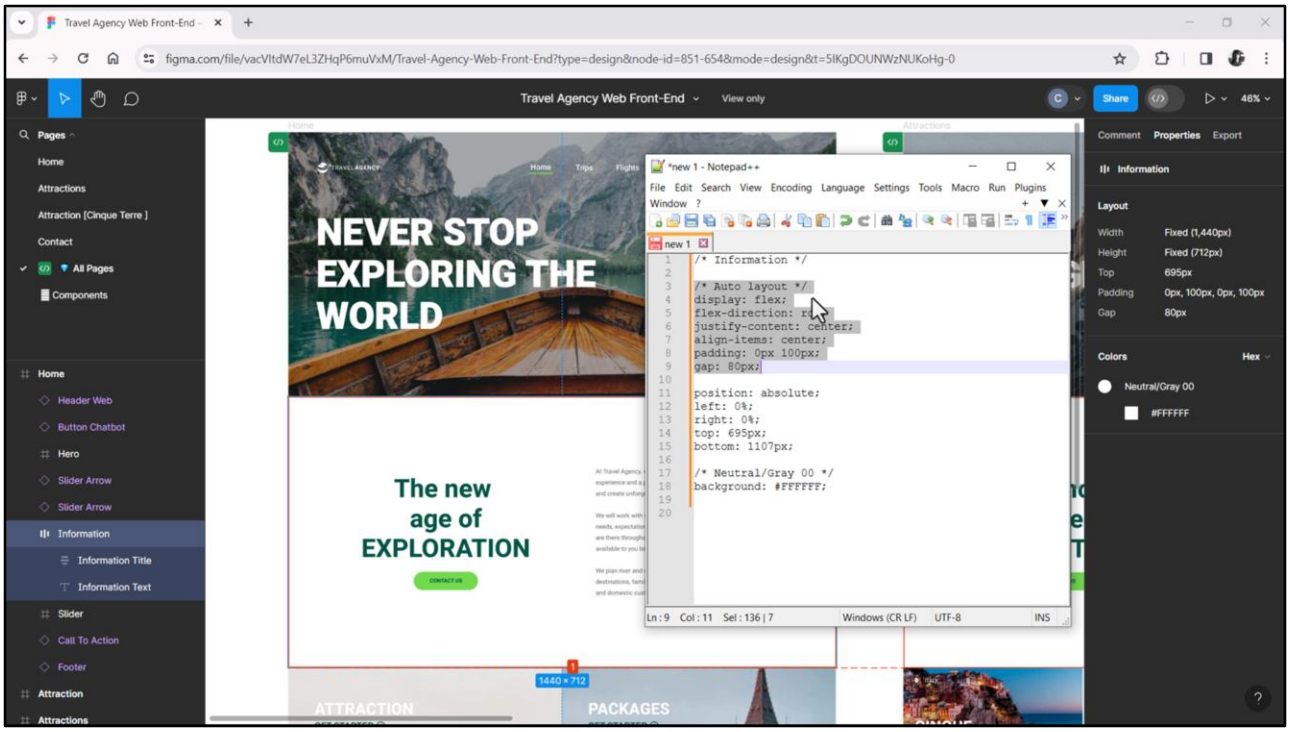
Neste caso, por exemplo, nos seria mais útil este controle antes do que o controle Table se quiséssemos, por exemplo, que quando a largura da tela diminua, se não couberem os dois elementos nessa largura, então apareçam em duas linhas (isto é, com Wrap). A tabela não nos permite isto, é muito mais estruturada: tem linhas e colunas. E em quantidades fixas.

Aqui temos apenas uma linha (se é que a direção é row, como esta) sem colunas, ou seja, onde os elementos são colocados um ao lado do outro conforme o espaço disponível. E se ficarem sem espaço, se tiverem a propriedade Wrap ativada, bem, então é criada uma segunda linha e assim sucessivamente até completar todos os itens.



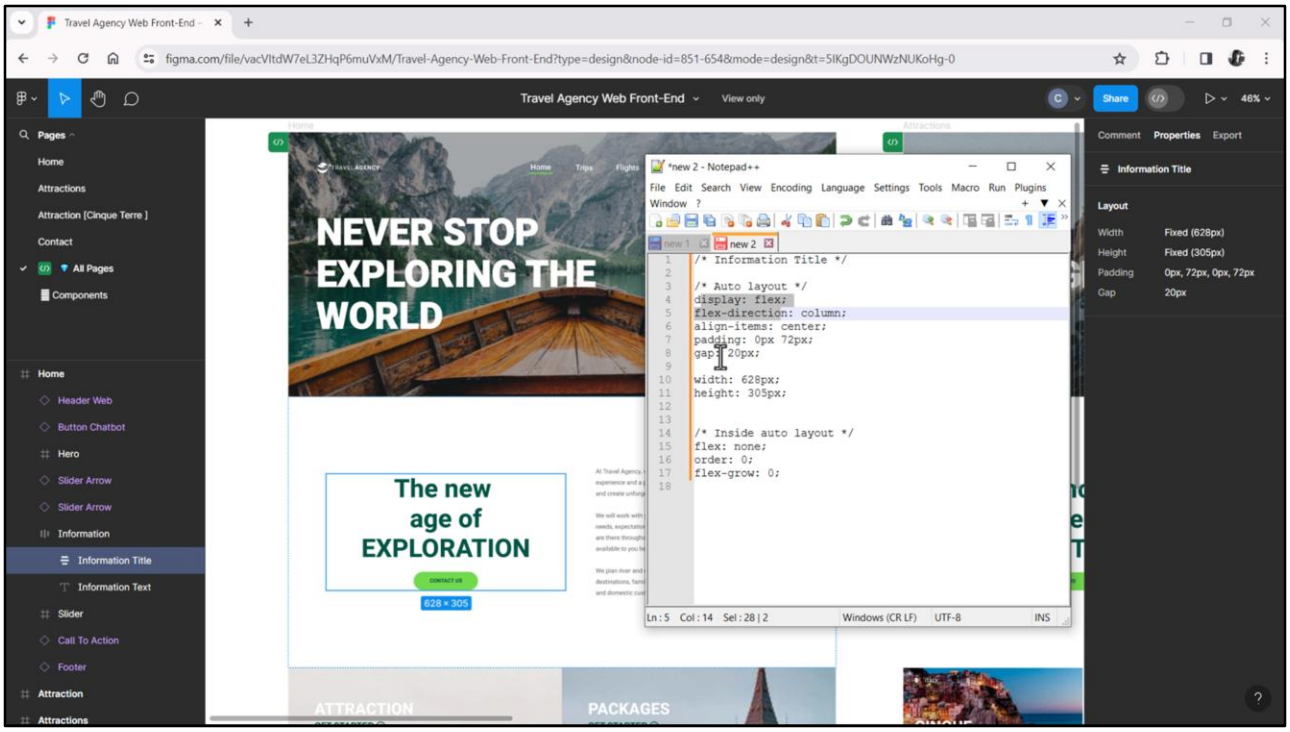
Quando um controle se encontra dentro de um container flex, adquire propriedades, estas aqui, justamente por essa situação.

É claro que, como com qualquer outro controle, podemos definir propriedades de design do flex, todas as que vimos de maneira estática, ou as mais conhecidas como margem, padding, etc., através de classes.



Por exemplo, voltando ao Figma, quando Chechu definiu este container com Auto Layout, determinou esta direção horizontal, e lhe apareceram para definir paddings em relação às quatro bordas: acima, direita, abaixo, esquerda. E também lhe apareceu a possibilidade de definir o Gap, que é a separação horizontal entre os elementos, que ela havia colocado que era de 80px.

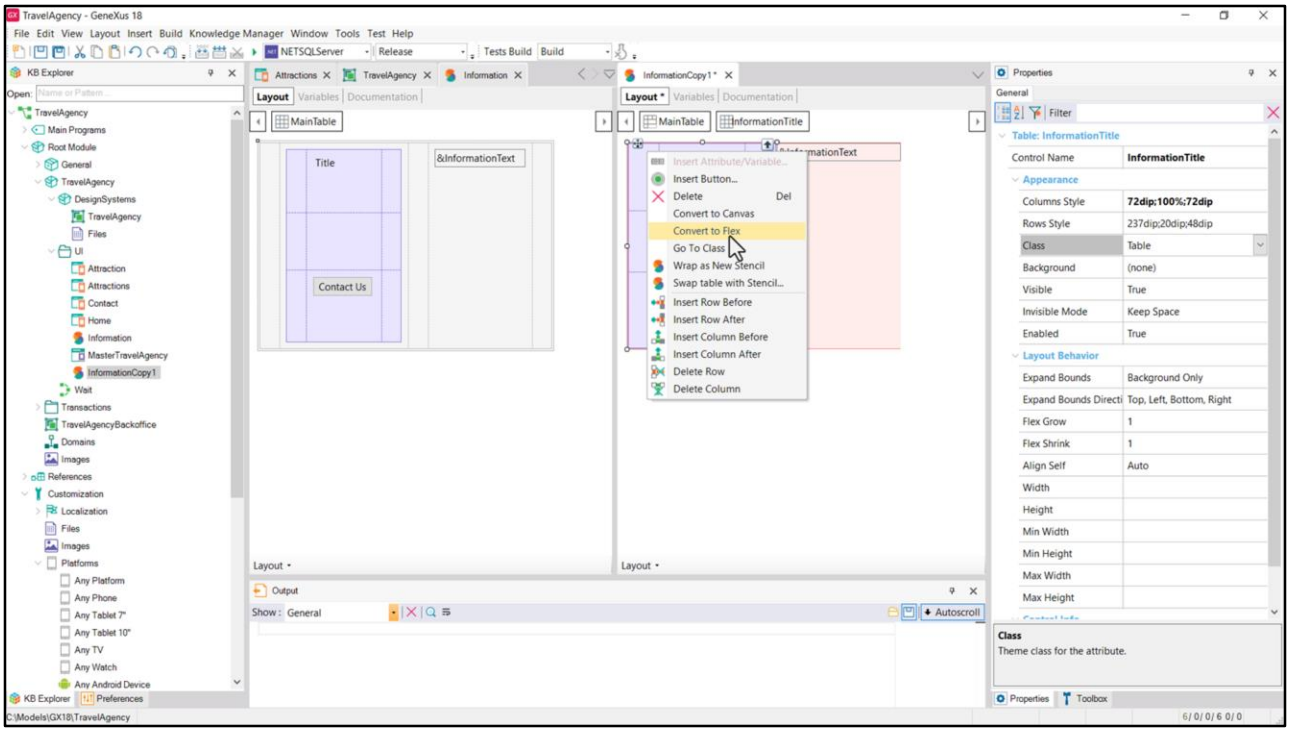
Vejam que código CSS nos aparece quando o solicitamos…
Tudo isso não lhe parece familiar? Display flex, flex-direction row, justify-content center, align-items center. Estas propriedades, indicamos estaticamente no nível de nosso controle flex, e estas duas, padding e gap poderíamos colocar em uma classe que associaremos posteriormente ao nosso controle flex. Ou também poderíamos colocar todas dentro dessa classe, porque todas aquelas que podemos definir no nível do controle também podemos defini-las no nível de propriedades de uma classe, certo?



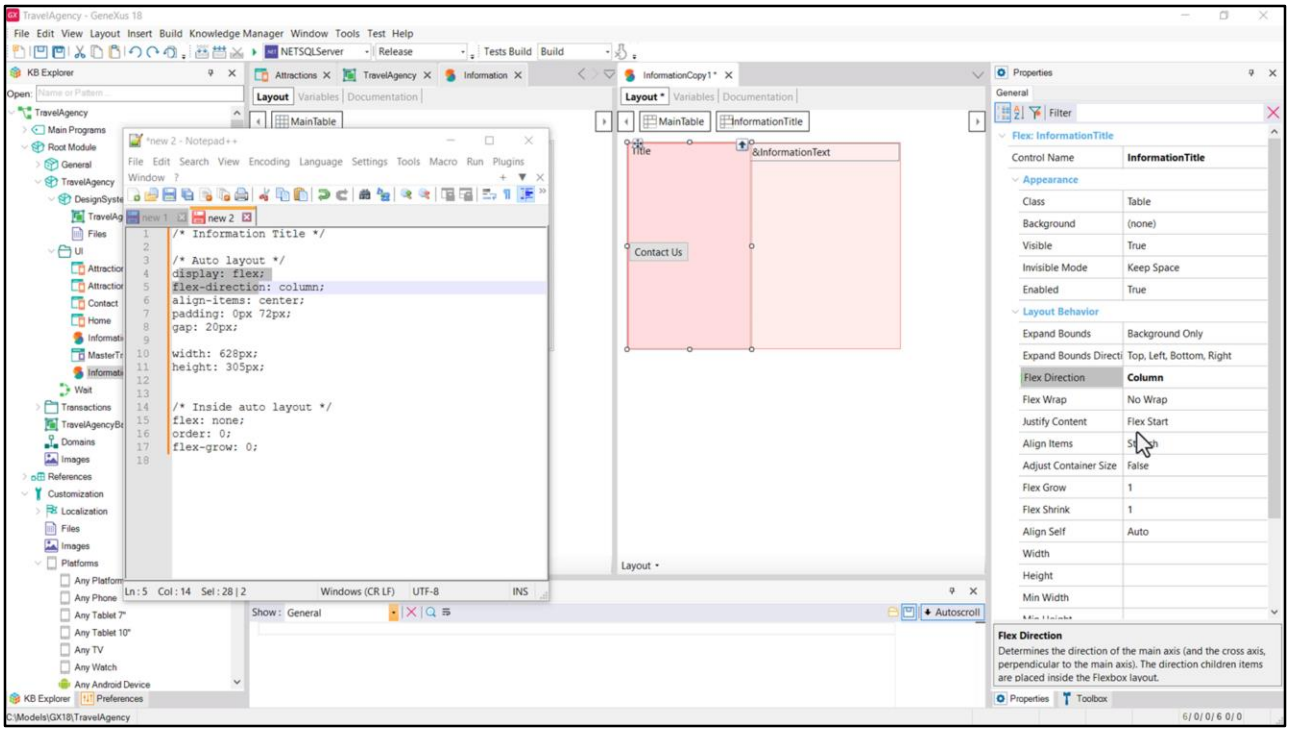
Bem, do mesmo modo, vemos que este outro container também tem Auto Layout, mas na outra direção, na direção column.

Aqui vemos as propriedades Padding e Gap.

E outra vez, se copiarmos as propriedades CSS, vemos aqui a display, a direction: column desta vez, a align-items e as propriedades de padding e gap.

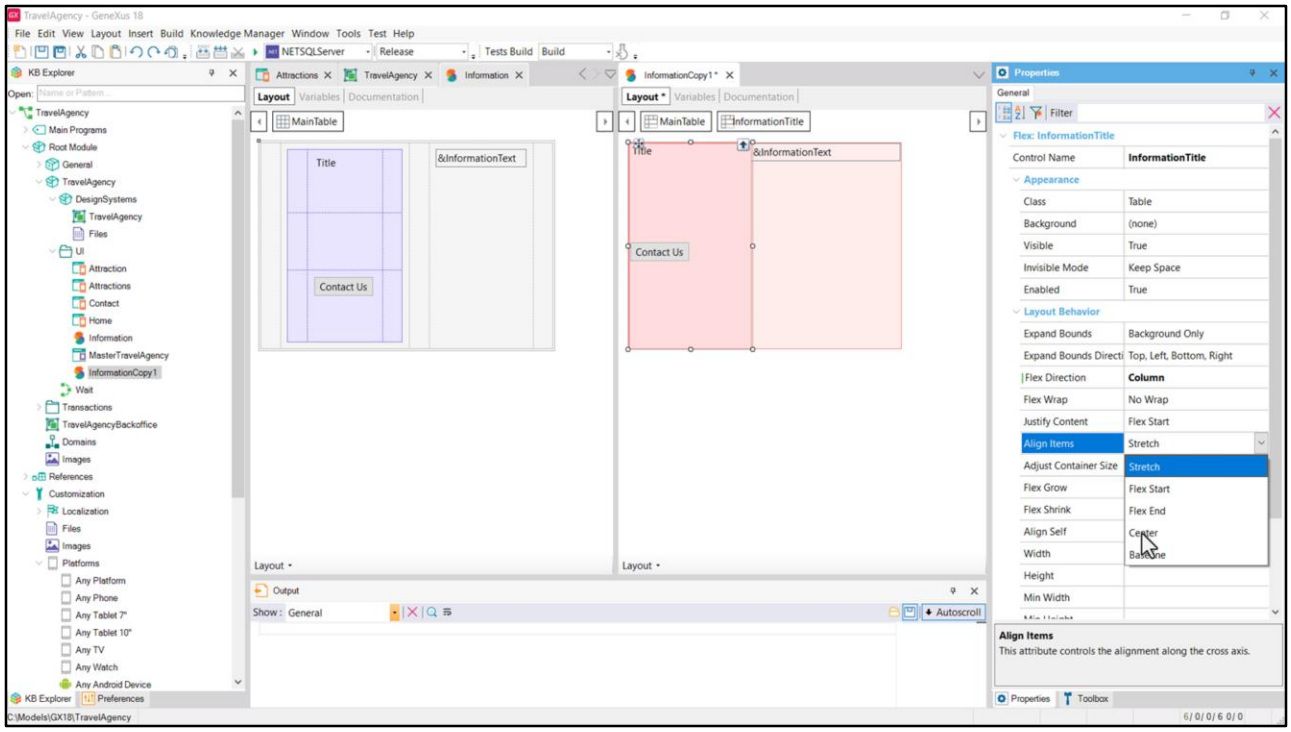


Então, poderíamos também definir esta tabela como Flex.

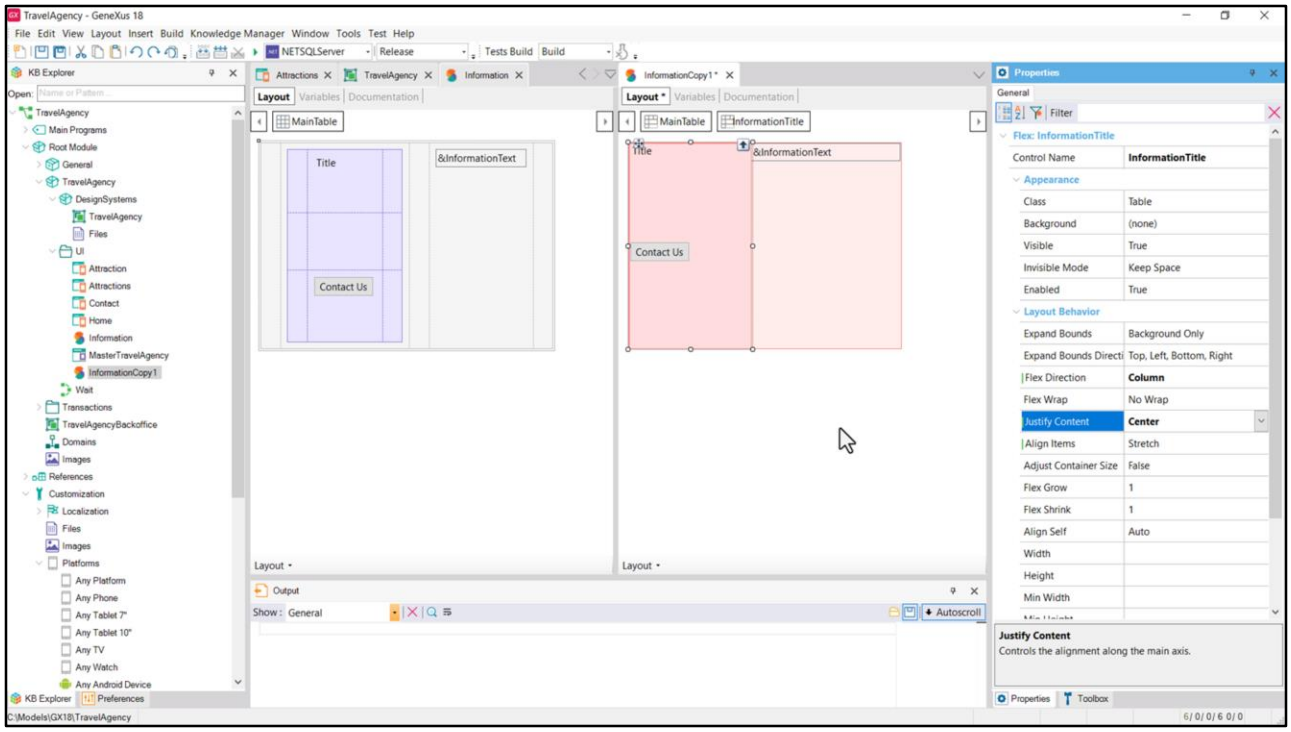


E colocar, claro, a direção column para que seja vista desta maneira.

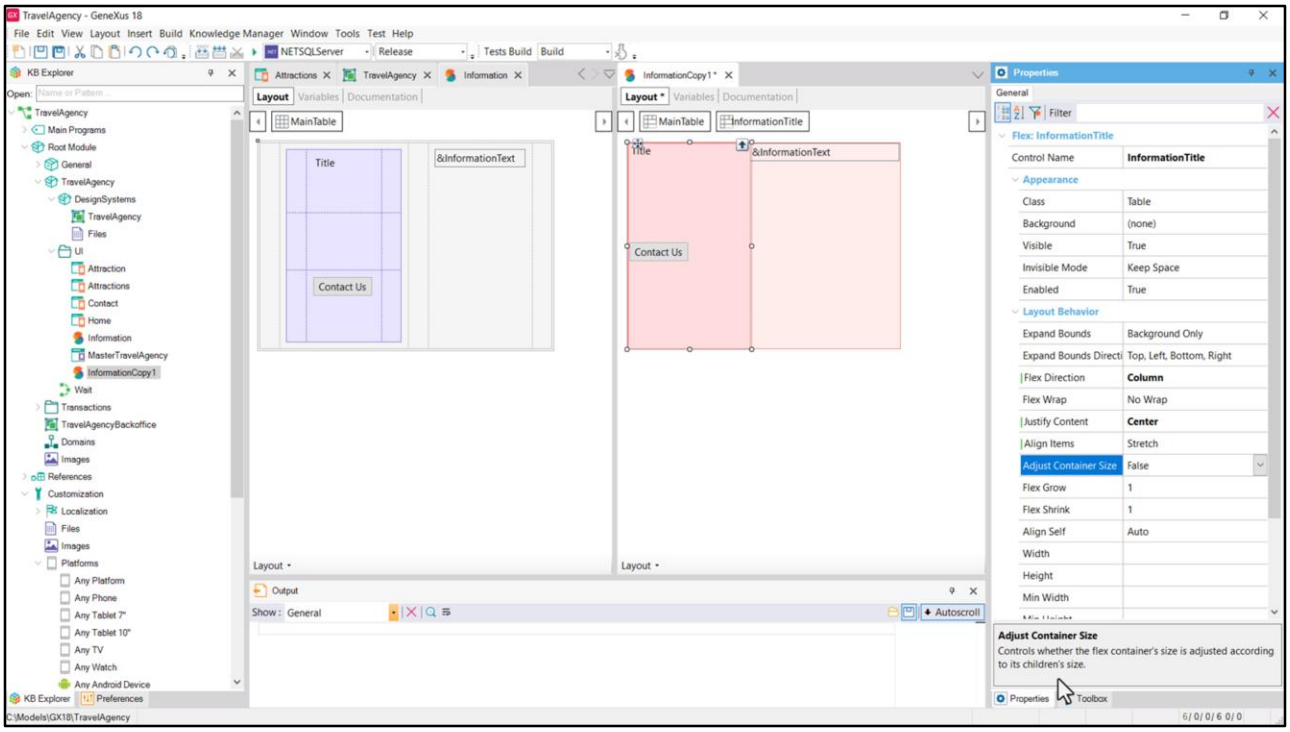
Do CSS de Figma também extraímos isto, que o align-items é centralizado, que significa estes itens em relação às bordas da outra direção, certo?



E bem, aqui então teríamos que escolher Center. Bom, estão desaparecendo, isto não deveria estar acontecendo... vou deixar o default, depois veremos.

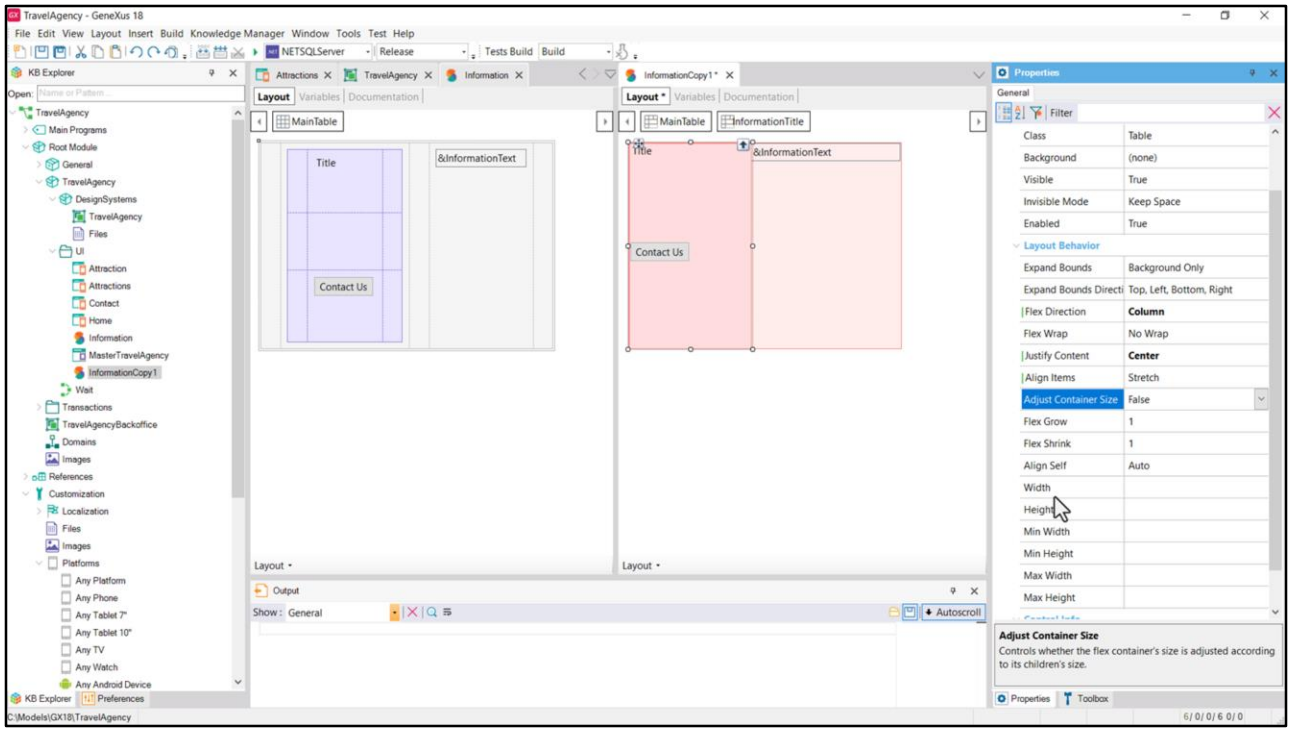


E por outro lado, a justificação do conteúdo tem a ver com a mesma direção do flex, certo?, que nesse caso é coluna, portanto também teríamos que dizer centralizado, certo? para o nosso caso, embora não nos tenha dito nada o CSS.

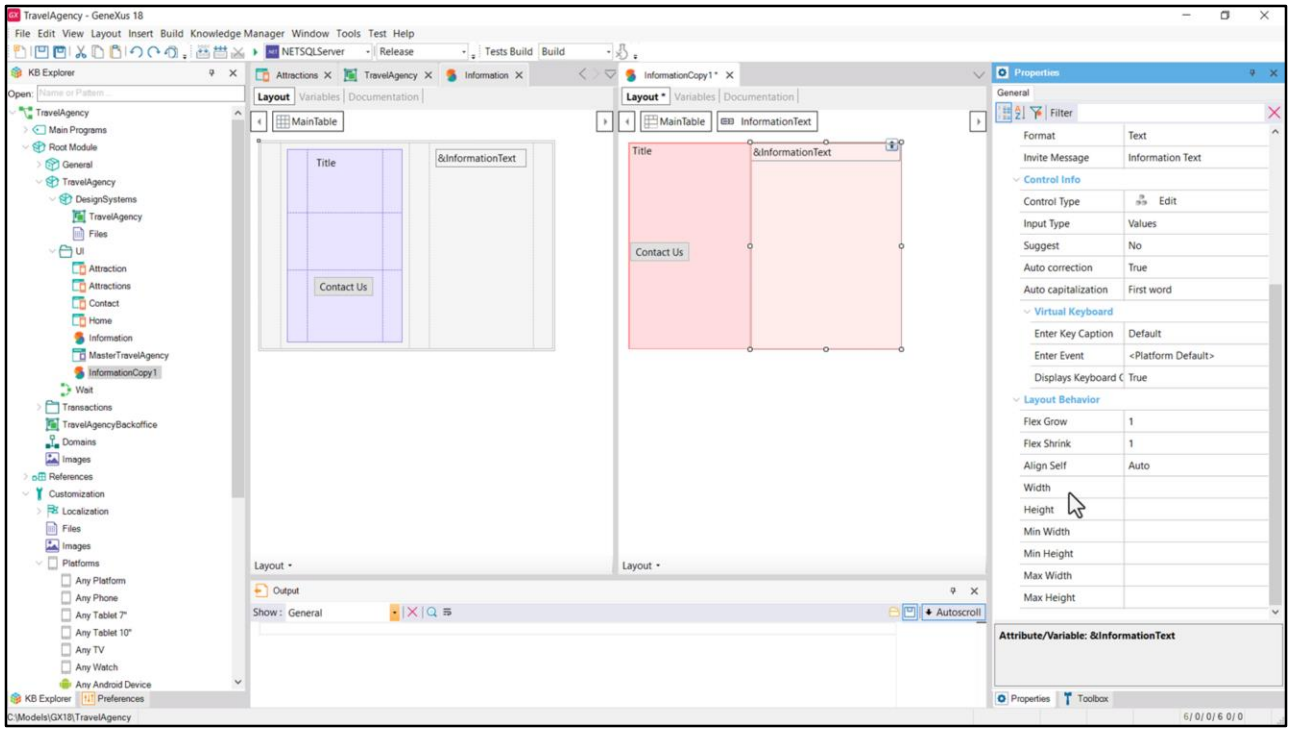


Por outro lado, como não tem wrap, não determinamos que tivesse Wrap, então esta propriedade atua e vemos que o que diz é que controla se o tamanho do container vai ser ajustado de acordo com o tamanho dos filhos.

Bom, então teríamos que jogar com essas questões para podermos posicionar nossos elementos de maneira correta de acordo com o design de Chechu. Pode parecer a princípio, mais simples utilizar os containers flex porque viram que podemos copiar diretamente esses critérios de design de Chechu... com o Auto Layout podemos meio que transferi-lo quase automaticamente para GeneXus. Mas temos que ter cuidado porque isto é um pouco mais complexo...



…veem todas as propriedades que estão aparecendo agora dentro desse grupo… nesse caso aparecem mais porque, estão aparecendo também as que fazem a participação deste flex interno dentro deste outro flex.



Como antes eu havia mostrado que aqui apareciam todas estas propriedades, de tamanho, por exemplo, de largura e de altura, que tinham a ver com a participação deste controle dentro do Flex pai.

Bom, por estas questões um pouco mais complexas é que preferi para começar, para ensiná-los, iniciar com tabela e não com flex. O flex pode parecer um pouco mais complexo por essas propriedades que vocês têm que começar a dominar melhor, às vezes se confundem, se estou falando de justificação do conteúdo, em que direção está, se tenho o alinhamento do item... então, eu começo a confundir, mas depois que você pega o jeito é bem simples. E bem, na realidade, tem que utilizá-lo com critério.

Quando utilizar o flex? Quando no design que estou implementando os elementos que estarão, por exemplo, ao longo de uma linha (se for essa a direção) não estão tão estruturalmente fixados em colunas, e, sem dúvida, claro, quando queremos a funcionalidade do wrap.

No nosso caso, a princípio não necessitávamos, porque já dissemos que quando a largura de tela for uma em que não cabem os dois elementos, na realidade será feito outro design diferente, que ainda não vimos porque pedi a Chechu que os desenhasse em outro arquivo, assim não nos confundiríamos agora.

Bom, por enquanto vou deixar por aqui, já foi introduzido, que era o que eu queria e logo mais adiante veremos se enfrentaremos a necessidade de utilizar um container flex em vez de uma tabela. Quando digo mais adiante, quero dizer quando continuarmos desenvolvendo as telas

de nossa aplicação.

Também um grid poderá ser flex. Já mencionei isso e voltarei a mencionar quando implementarmos um dos grids que temos pela frente.

Com isso vou encerrar o vídeo porque quero passar para o próximo módulo, mas vocês podem, se quiserem, testar o que eu não testei, atribuir classes aos flex com as propriedades correspondentes, e prestar atenção nos tamanhos que vão dar aos controles, analisar um pouco e tentar fazer que a aplicação com o stencil com flex pareça igual à da outra maneira, como a tínhamos feito. Bom, e esse é o desafio que deixo para vocês.

Nos vemos no próximo módulo.


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