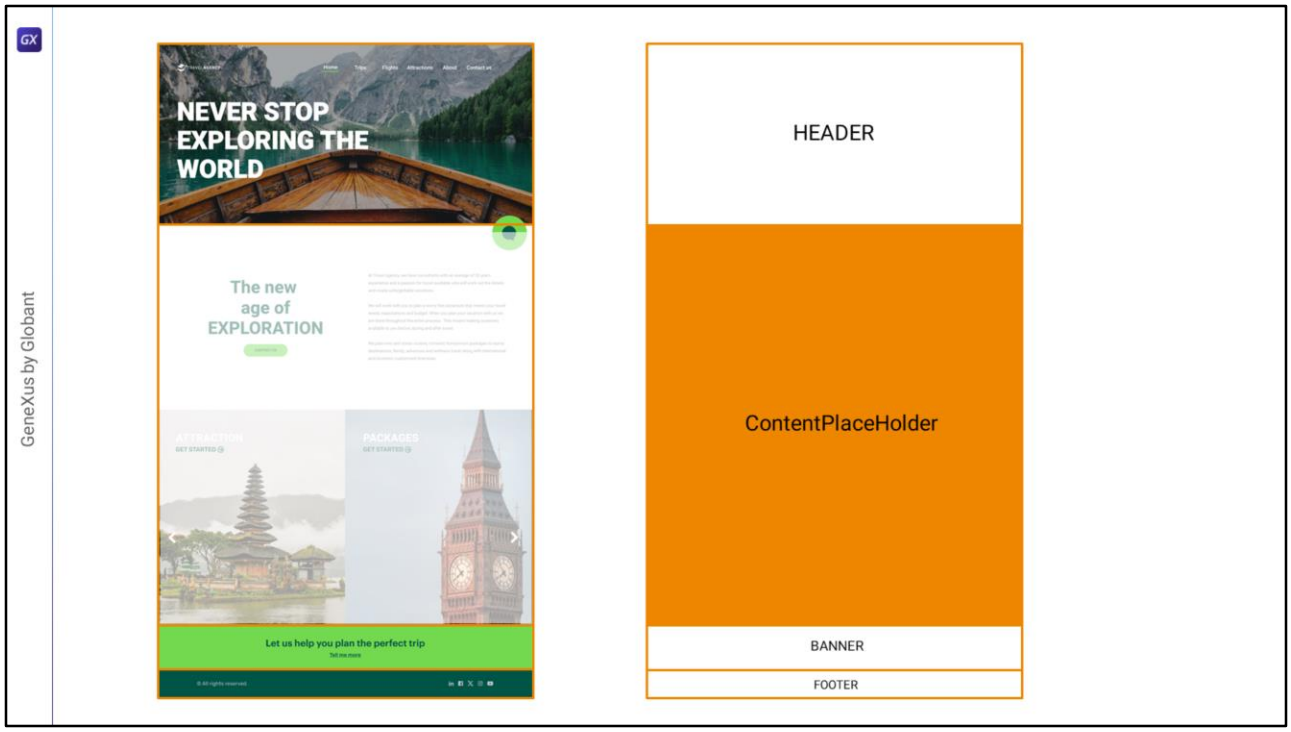
GeneXus by Globant

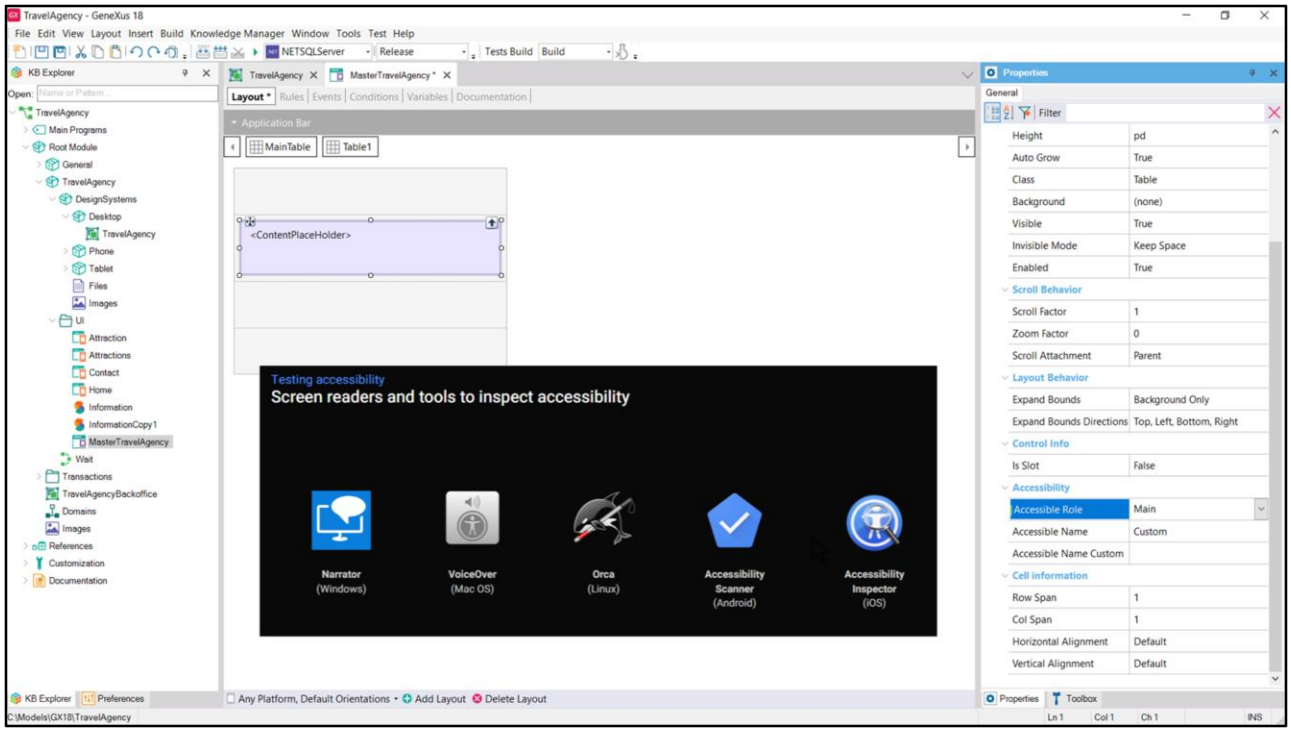
Accessibility and first steps for Master Panel

Cecilia Fernández



Aqui temos a primeira tela da aplicação, da qual extrairemos o que corresponde ao Master Panel. A área central é aquela que corresponderá ao espaço dentro do qual serão carregados os layouts de cada um de nossos painéis individuais: o panel Home, o de Attractions, o de Attraction, o de contato.

Portanto, a estrutura do Master Panel poderia ficar desta maneira: como uma tabela com 4 linhas; na primeira das quais implementaremos o Header; a segunda conterá o controle ContentPlaceHolder, que é justamente o controle que faz com que as páginas individuais que tenham esse como seu Master Panel sejam carregadas ali dentro; para a terceira linha teremos a implementação do banner; e na quarta implementaremos o footer.

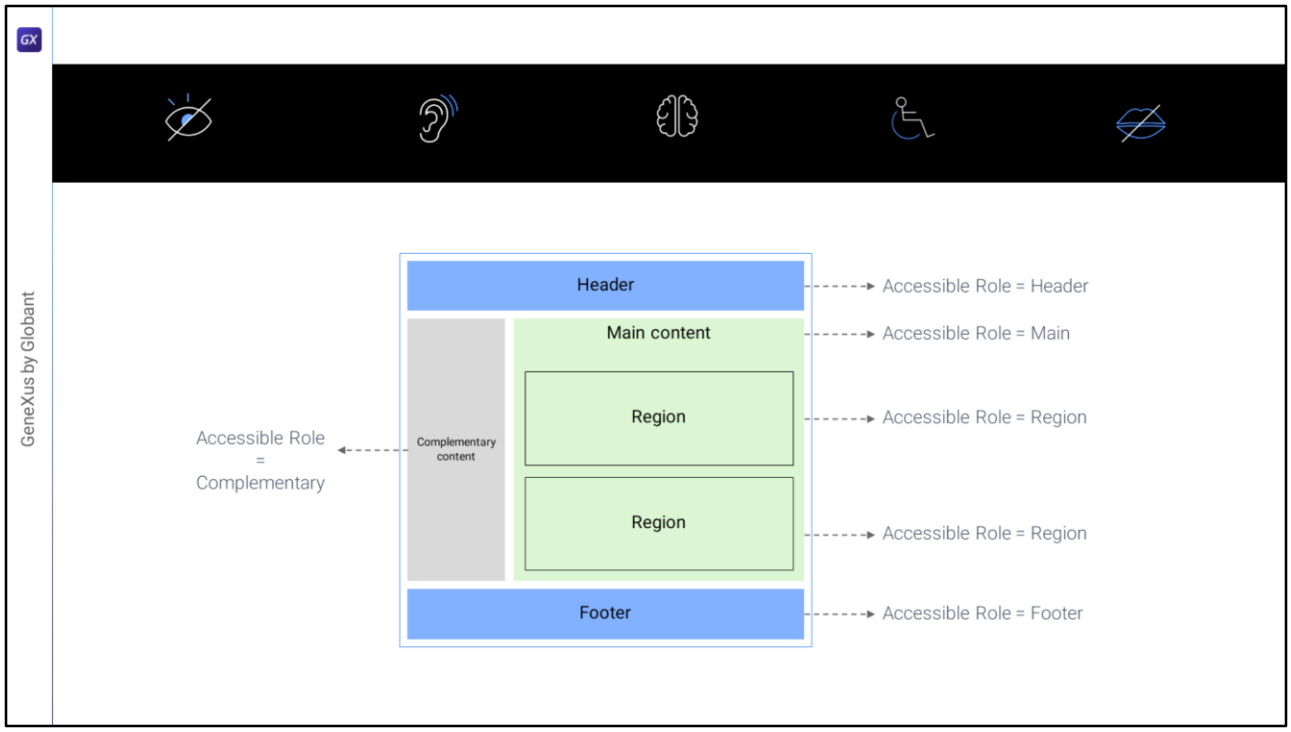


Então vamos começar por esta estrutura em nosso objeto MasterPanel.

Tínhamos apenas o ContentPlaceHolder, mas agora vamos colocá-lo dentro de uma tabela, e ambos dentro de outra tabela que terá 4 linhas.

Por que colocamos o ContentPlacerHolder dentro de uma tabela e não simplesmente dentro da célula da linha 2? Porque a tabela nos permitirá semantizar seu conteúdo: podemos dizer que aqui estará o conteúdo principal da página. Para quê? Para tornar a aplicação mais acessível. Por exemplo, se um usuário tiver dificuldades visuais, poderá utilizar um dos muitos programas leitores de tela que poderão informar por áudio que esta parte corresponde ao conteúdo principal.

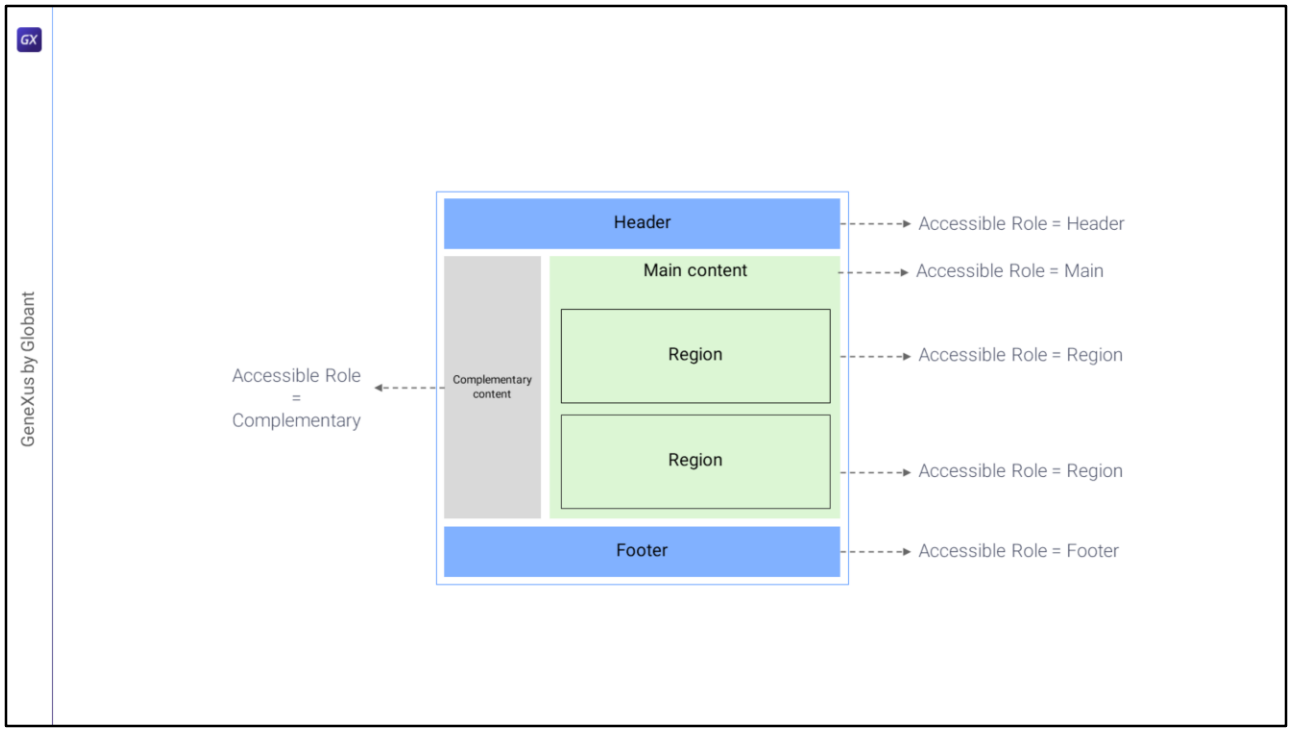
Vamos observar as opções de acessibilidade que aparecem no nível da tabela. São estas 3: a 2 e a 3 vão juntas, como veremos.
A primeira, a da role, pode ter algum destes valores.



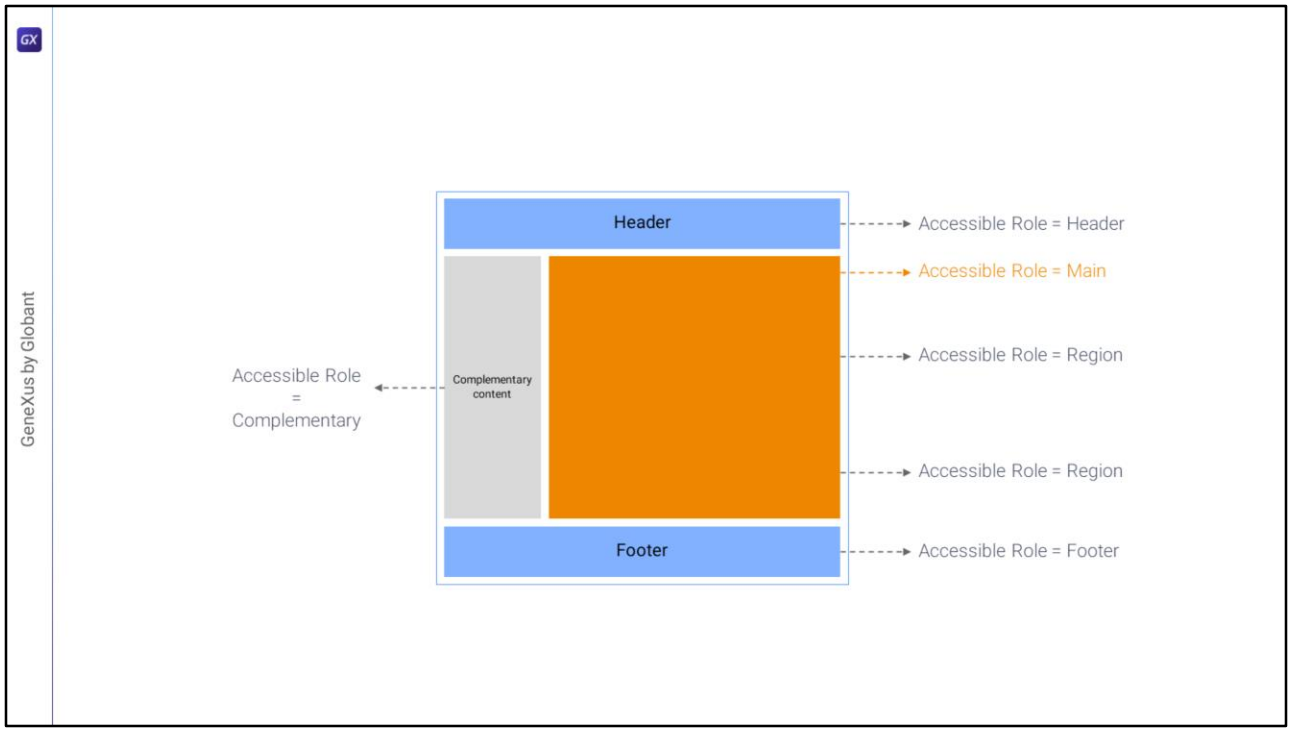
Aqui vemos a semântica da maioria deles.

A acessibilidade de nossas aplicações é cada vez mais importante, sendo inclusive exigida por lei em muitos casos, em muitos países, e não beneficia apenas pessoas com limitações físicas ou cognitivas, mas também quando há limitações com a velocidade de internet, por exemplo, porque o usuário poderá saber qual conteúdo vai ali mesmo que tenha que esperar para visualizá-lo. Além disso, melhora a otimização de motores de busca (o que é conhecido como SEO) e então a gente encontra nossa aplicação web com maior facilidade.

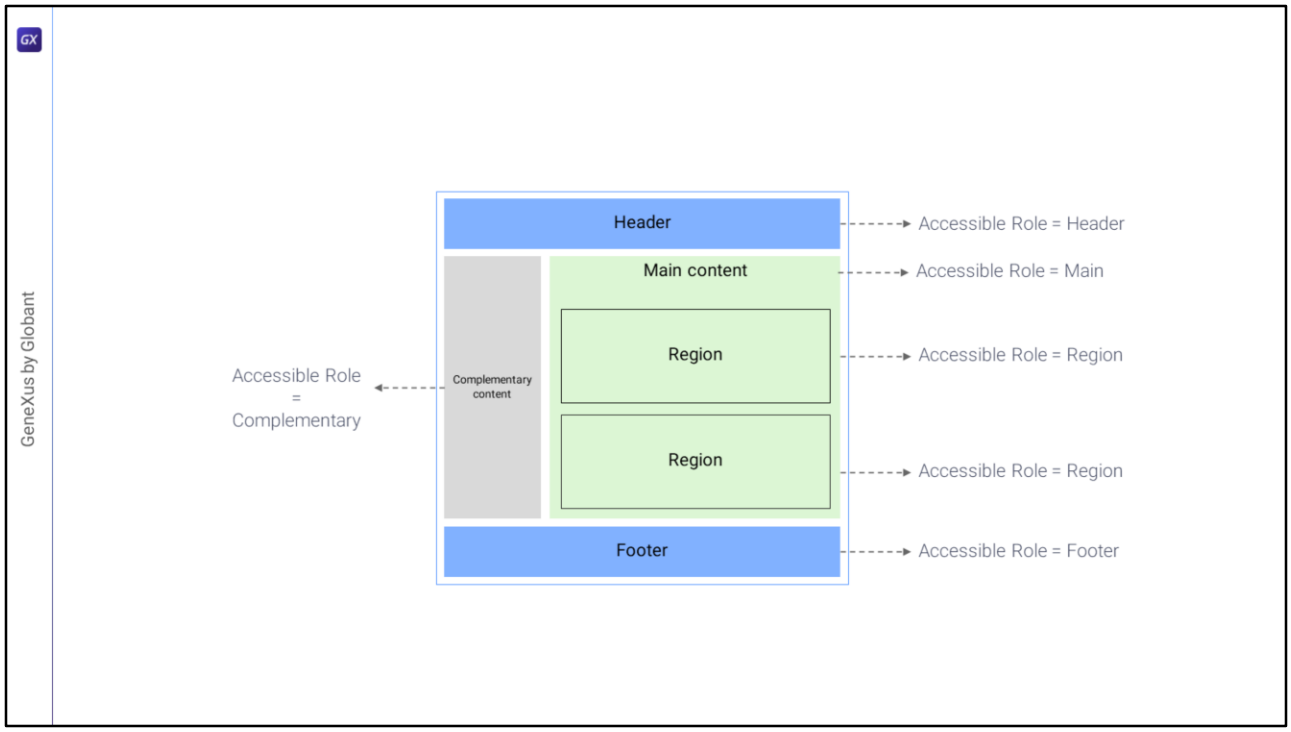
Se desde o início do desenvolvimento estivermos atentos para tornar a aplicação o mais acessível possível, o custo é mínimo e o benefício é alto. Ter que refazer o desenvolvimento posteriormente para atender à acessibilidade terá um custo muito maior. Então vamos atender a isto desde o começo!



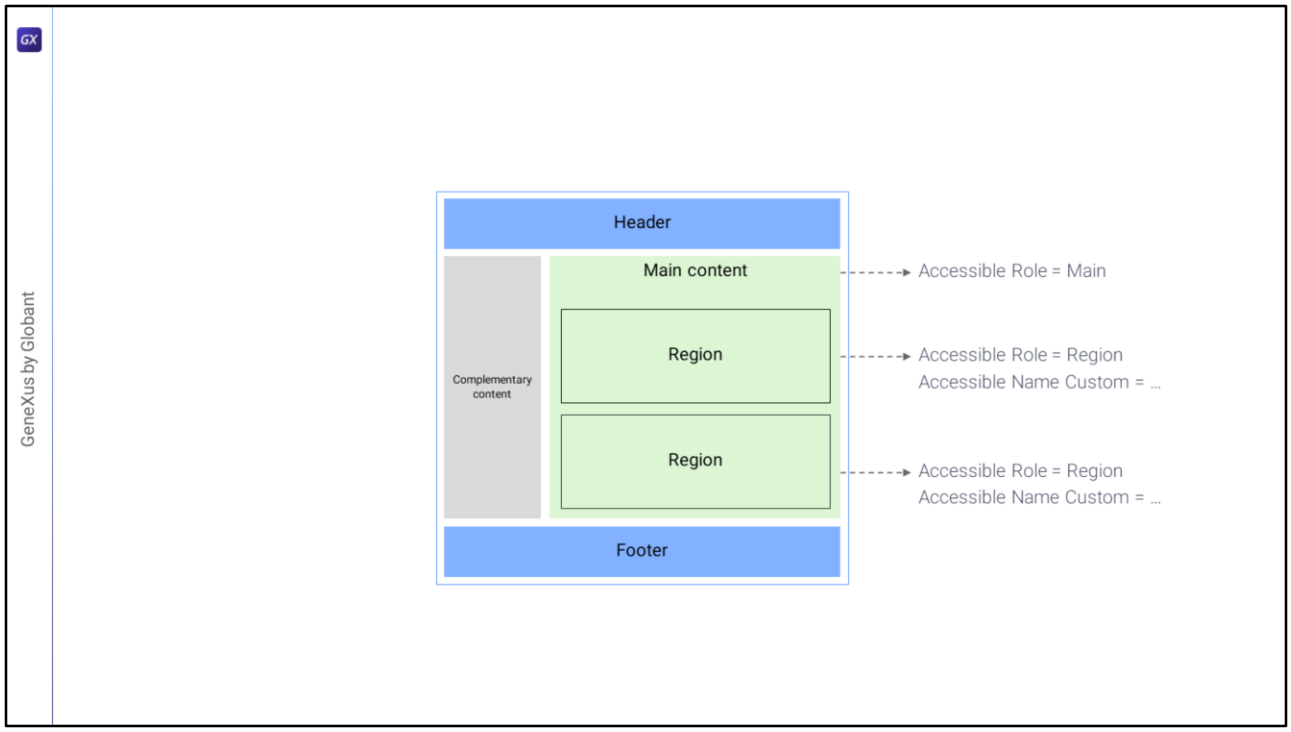
É evidente, então, que teremos que construir o layout de forma que nos permita indicar as **roles dos containers…**



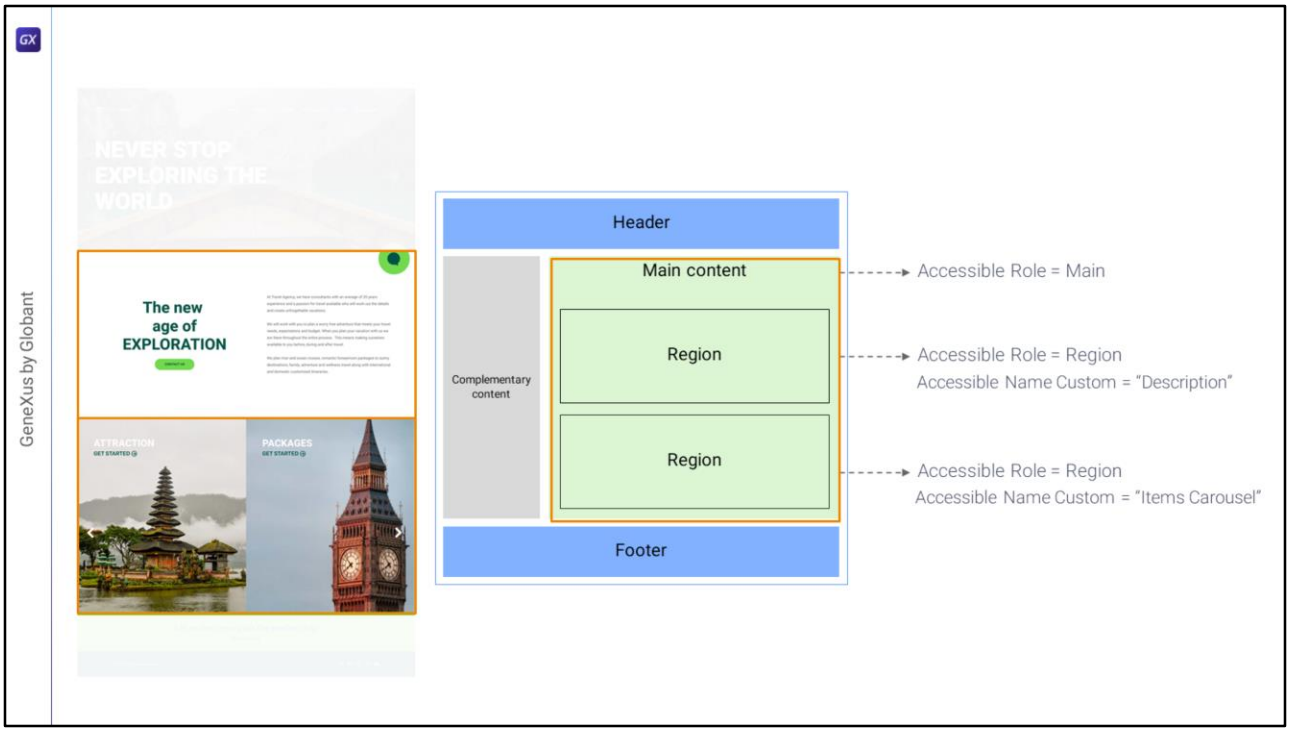
(por isso precisamos colocar dentro de uma tabela o ContentPlaceHolder, porque apenas os controles tabela (em todas as suas variantes, ou seja, também flex e canvas) têm a propriedade Role).



E na verdade essa propriedade só terá efeito para Angular. No momento, a aplicação nativa não faz nada com esta informação.

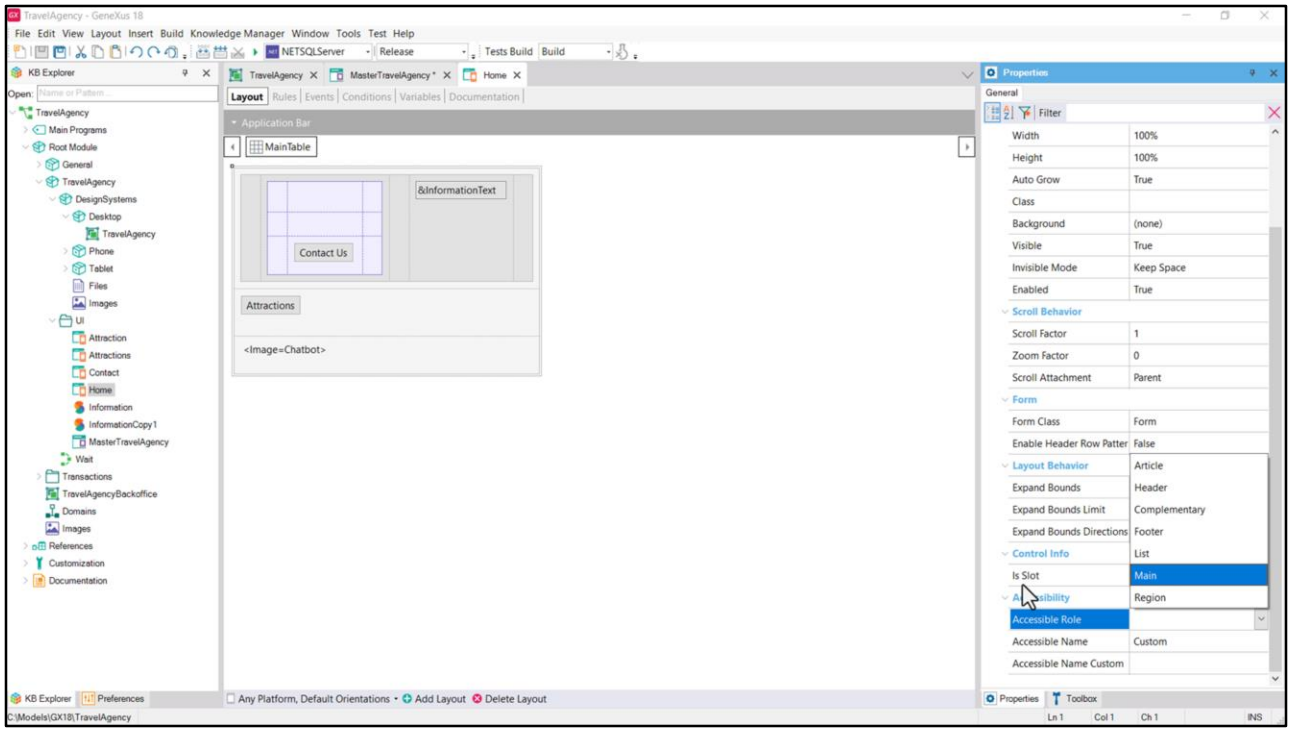


Por outro lado, embora por página possa haver apenas um container Main, poderá haver muitas regiões, então, para descrevê-las semanticamente, precisaremos de outra propriedade, que será a Accessible Name Custom.

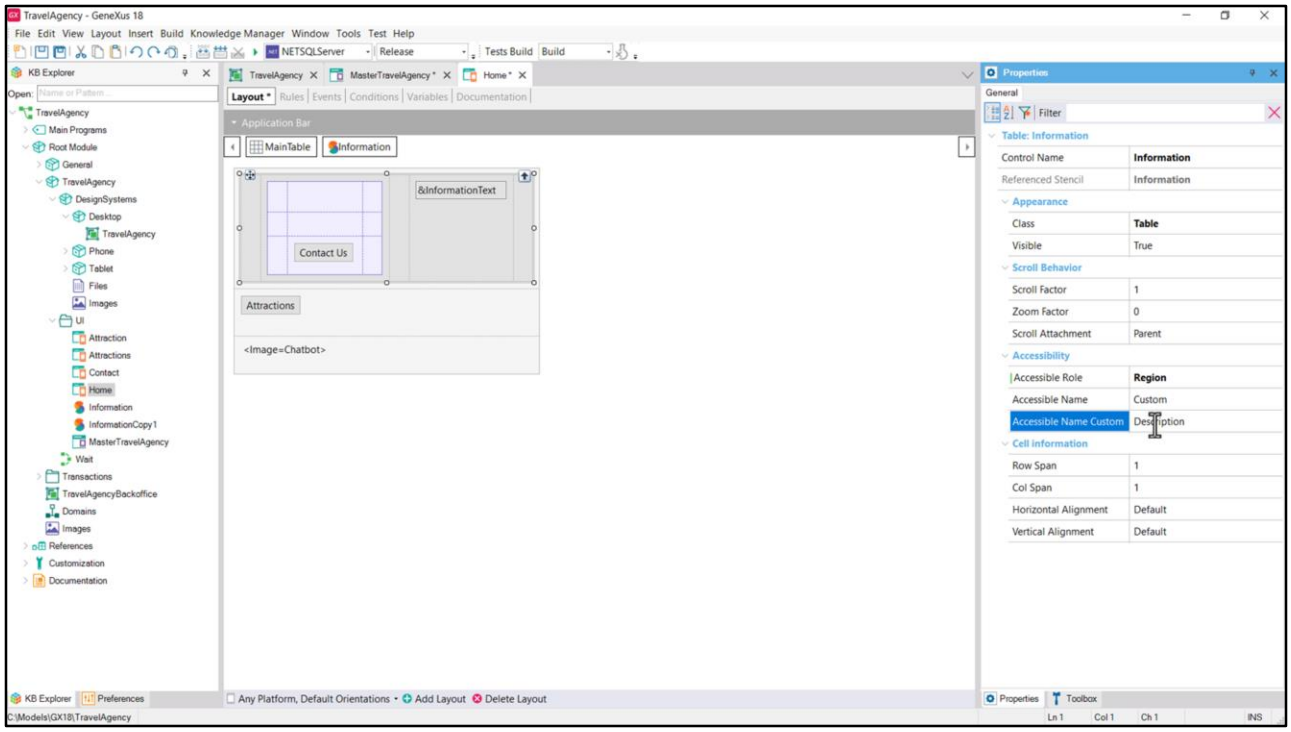


Se pensarmos no panel Home (e não no Master Panel): teremos uma tabela principal, Main, com duas linhas, e dentro de cada uma delas uma tabela com Accessible Role Region e Accessible Name Custom. E o nome descritivo do conteúdo da primeira região poderia ser “Description”, e o da segunda “Items Carousel” para indicar que ali haverá um carrossel de itens.

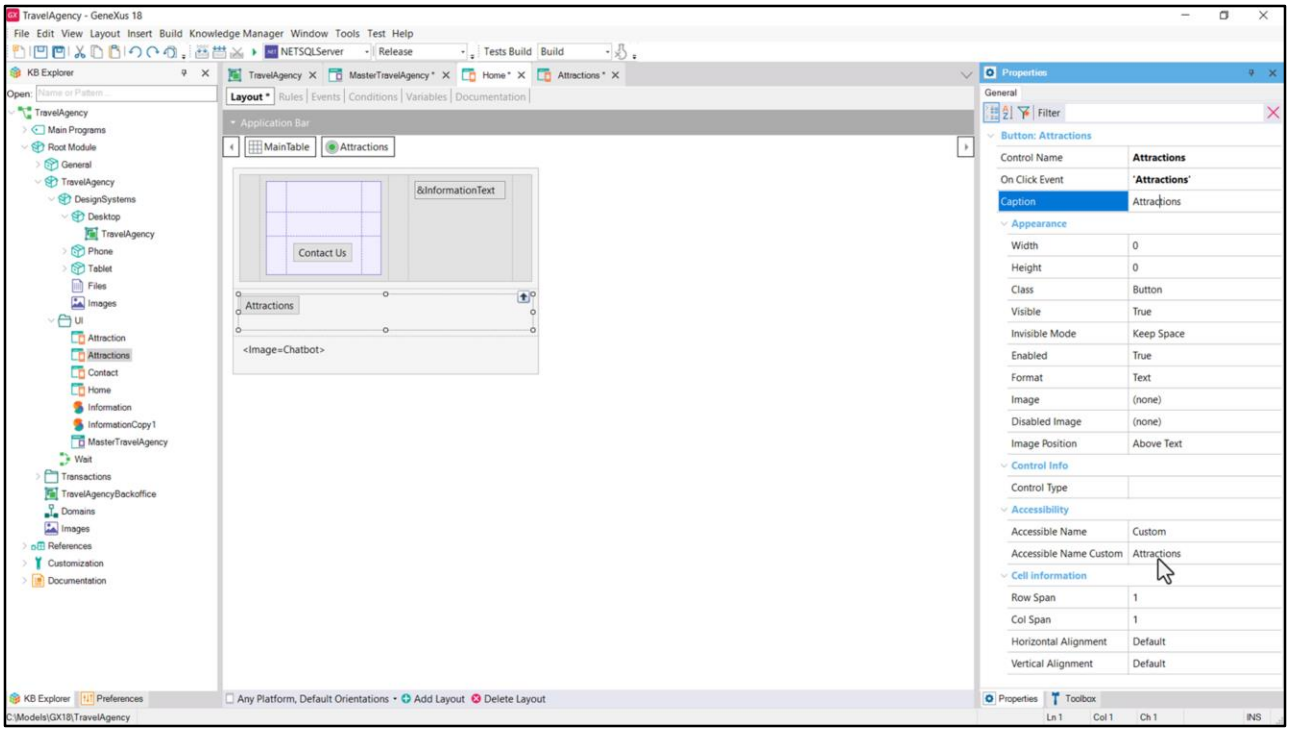
Então vamos para a página Home…



… para a tabela main definiríamos como Role a Main…



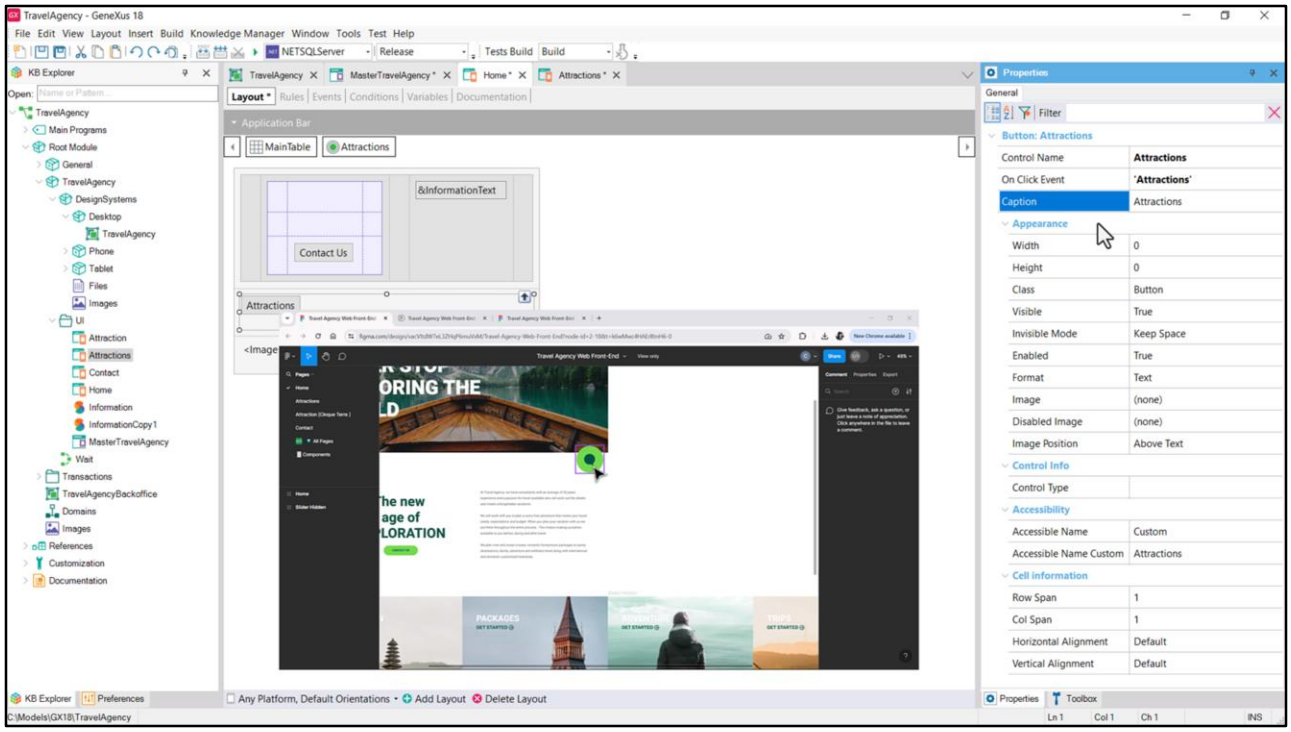
…e para a tabela da linha 2 colocaríamos Region, para o Accesible Name deixamos o valor Custom (porque o outro valor possível é para que pegue a descrição a partir do caption de um text block, que poderia muito bem ser neste caso, mas veremos depois) e colocaríamos como descrição então, nesta propriedade o valor Description. Faremos o mesmo no panel Attractions.



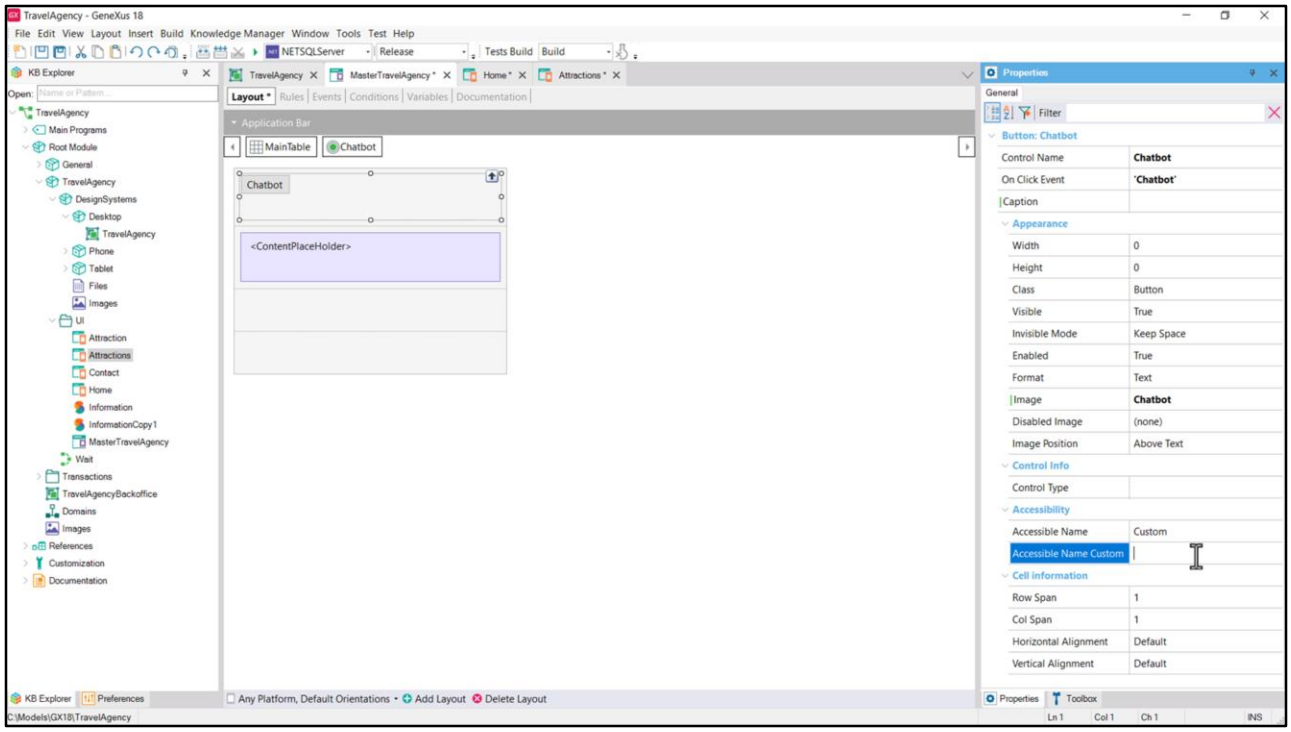
Ainda não implementamos o carrossel para a Home e já sabemos que esse botão está aqui temporariamente, vamos removê-lo quando implementarmos o menu. E que esta imagem tínhamos deixado apenas para mostrar algo sobre as imagens em um vídeo anterior, ela também não permanecerá.

Mas vamos aproveitar para observar que ambos os controles possuem propriedades de acessibilidade. Para o botão que possui Caption Attractions, observemos que por padrão GeneXus colocou esse mesmo valor para a descrição semântica do botão.

Que seja um **botão** (e não um textblock com evento associado, por exemplo) e que sua descrição semântica seja Attractions terá algumas consequências importantes na aplicação gerada para Angular: por um lado, por ser um botão, já tem a semântica universal do botão: então poderá ser manipulado por teclado, focando-o com o tab e acionando com Enter, por exemplo. E por ser botão e ter descrição, um leitor de tela informará ao usuário que se trata de um botão com descrição Attractions.
Aqui não tivemos que fazer nada porque o botão assumiu como Accessible Name Custom o Caption do botão.

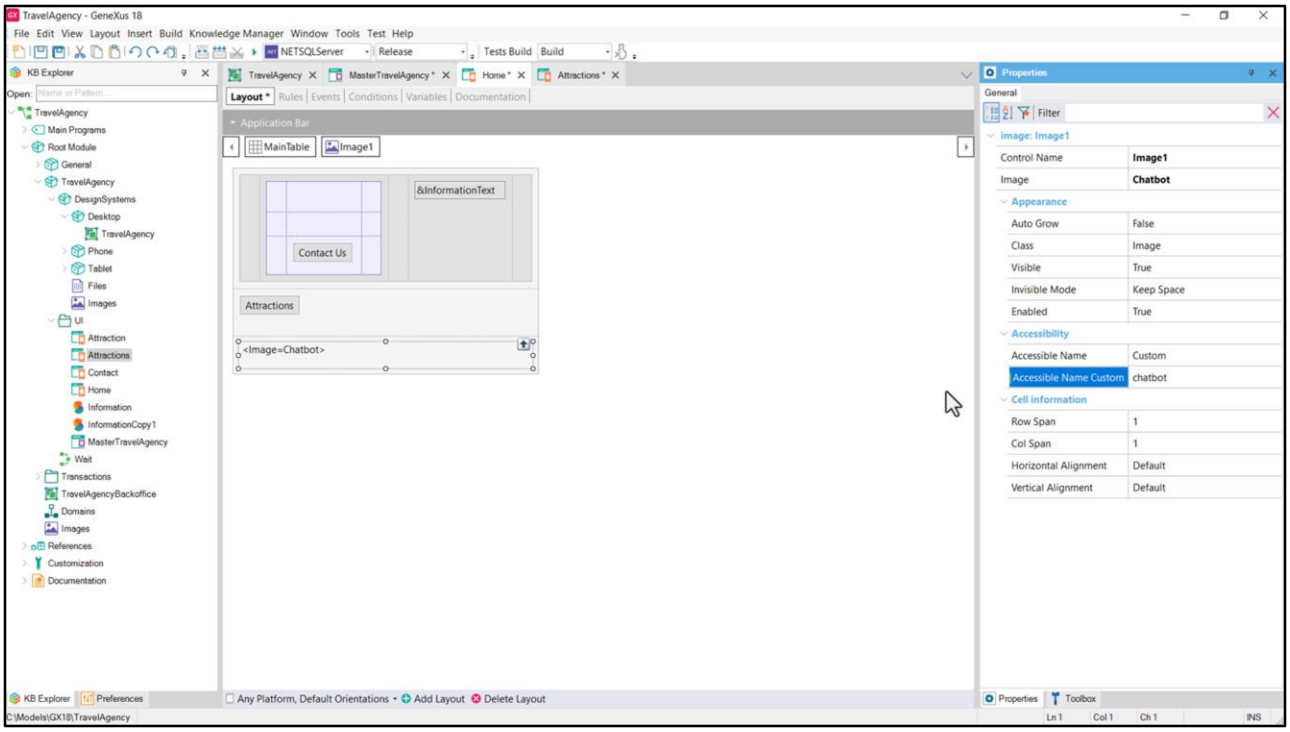


Mas se não tivesse Caption, por exemplo porque vai ser mostrado com uma imagem unicamente, como será o caso da imagem do chatbot…

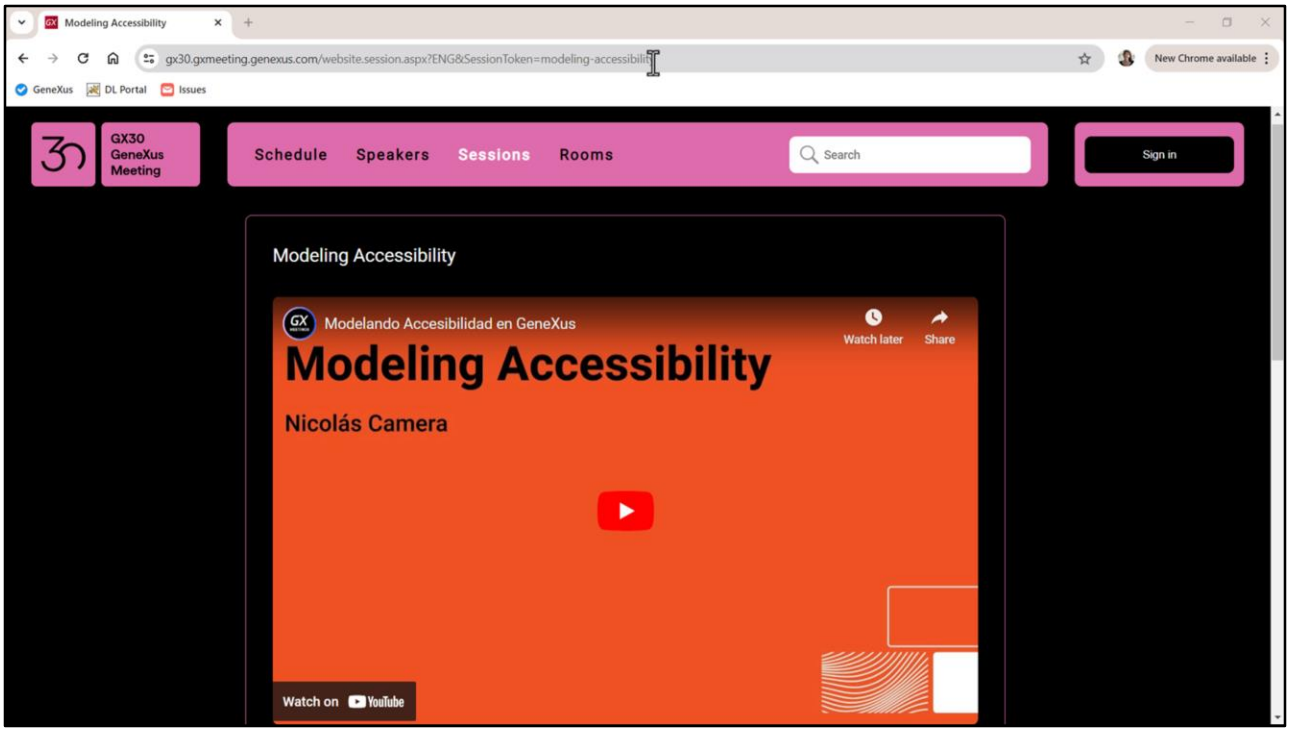


…e já podemos fazer isso no Master Panel para mostrar...

…arrasto o controle botão… associo um nome ao evento que será disparado quando for pressionado… esse nome é o que por default é associado ao Caption… mas não quero que o usuário veja um texto e sim a imagem do chatbot… então eu associo a imagem que já tinha carregado na KB e deixo vazio o Caption…
Mas ao fazer isso, a propriedade Accesible Name Custom também fica vazia. Aqui é onde temos que intervir explicitamente. Colocarei “Chatbot”, como sua semântica.

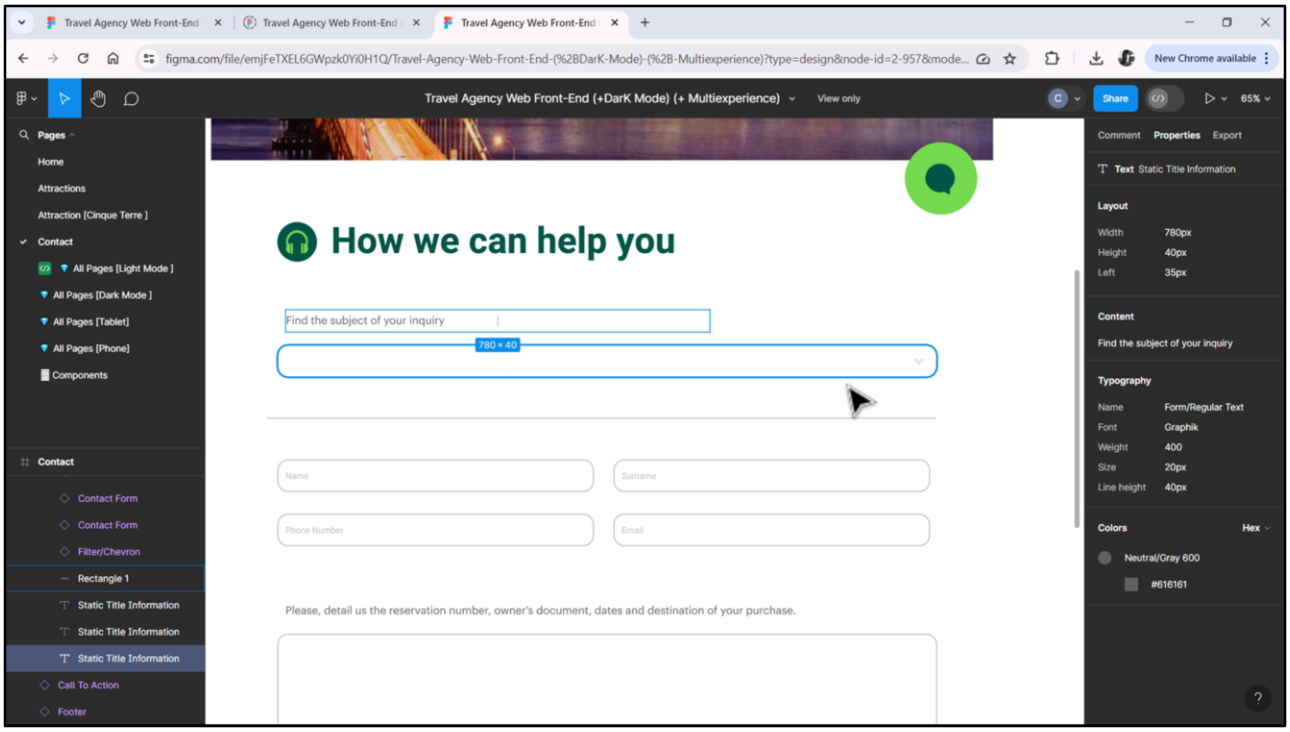


Podemos ver o mesmo com o controle imagem inserido em um layout, como imagem e não como botão. Por padrão, um controle de tipo imagem não tem nenhuma propriedade descritiva, portanto, não há nenhum valor default a ser fornecido aqui. Teremos que escrever explicitamente.



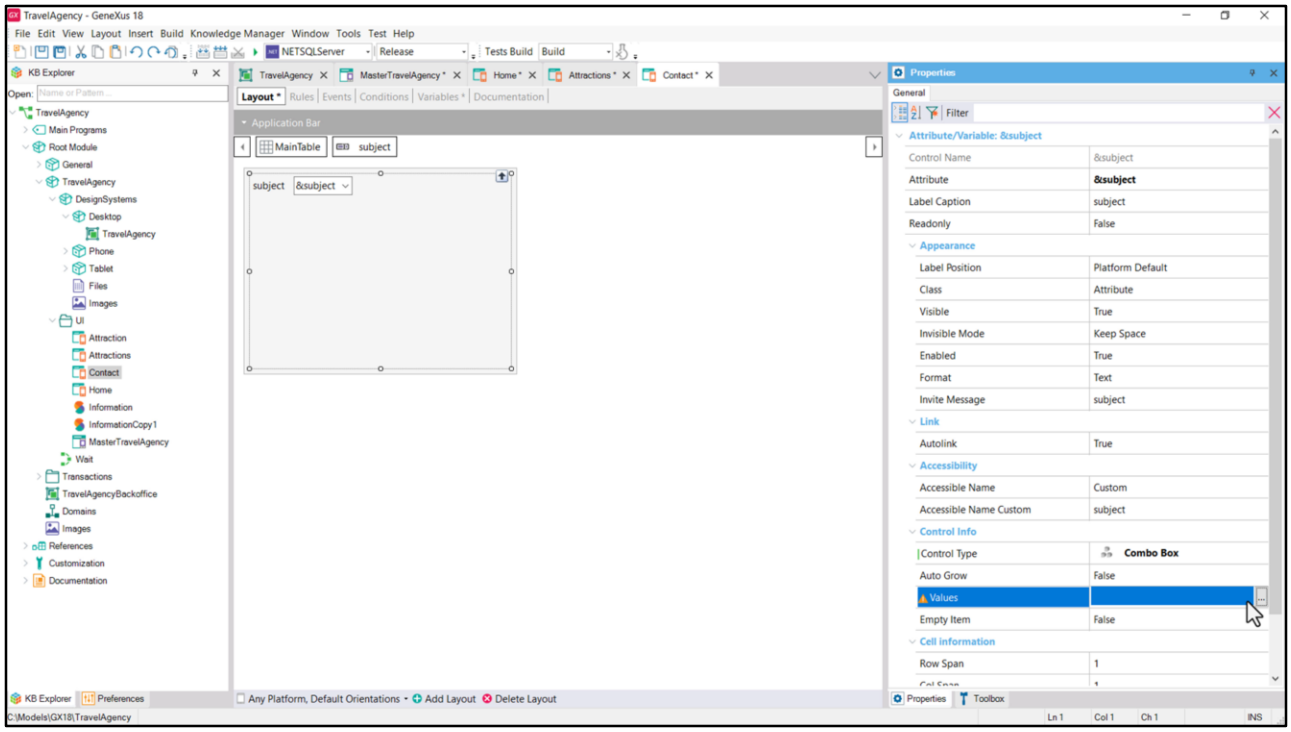
Aqui poderão ver uma apresentação do final de 2023 onde Nico Cámera analisou todos os aspectos de acessibilidade a serem levados em consideração e forneceu dicas para atendê-la desde o início do desenvolvimento com GeneXus.

Ali vocês verão que GeneXus já faz muita coisa por nós sem que tenhamos que nos preocupar com nada além de escolher os controles apropriados e fornecer, por meio de suas propriedades, valores semânticos descritivos.

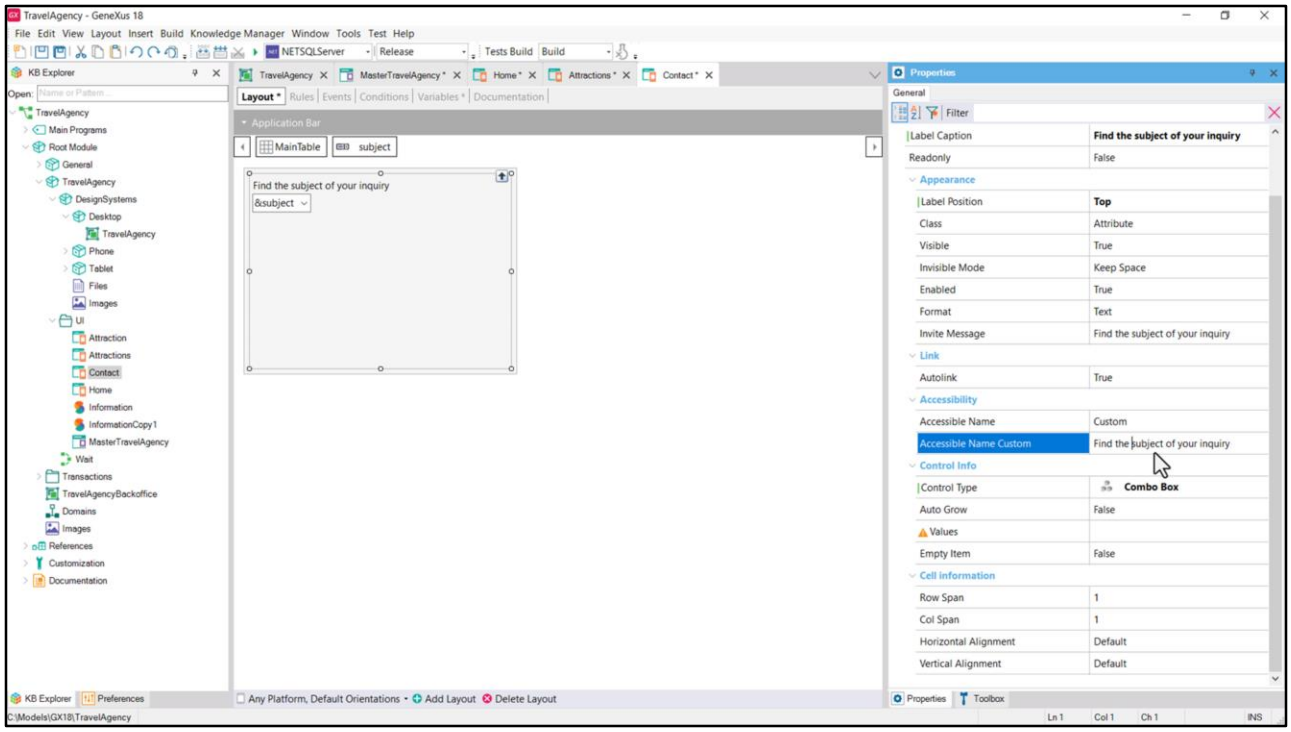


Assim, por exemplo, nosso formulário de Contato terá campos a serem preenchidos pelo usuário.

O primeiro será um combo-box, e sua função semântica é dada por este texto... aqui vemos que se trata de um combo.

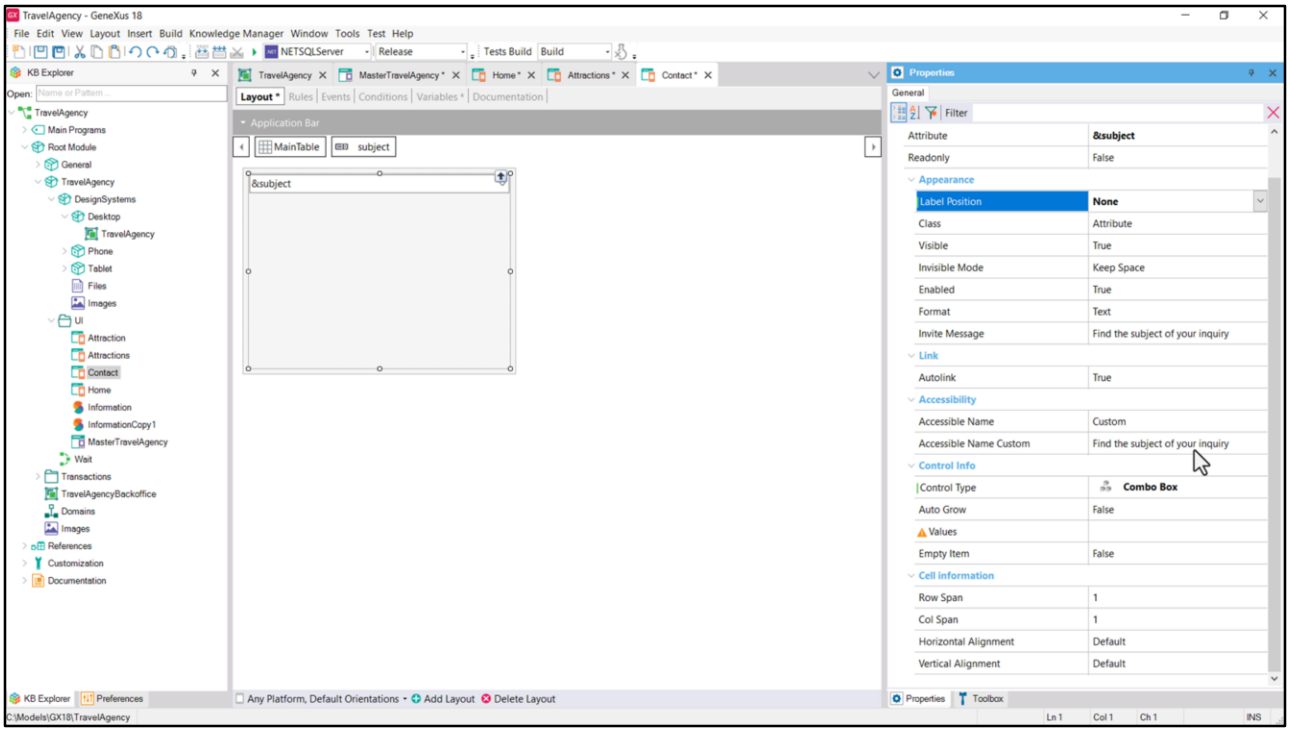


Então, implementaremos isso com uma variável, que chamei assim, que inseriremos no layout através de um controle attribute/variable ao qual especificaremos como tipo de controle o combo box. Ali devemos inserir suas opções.

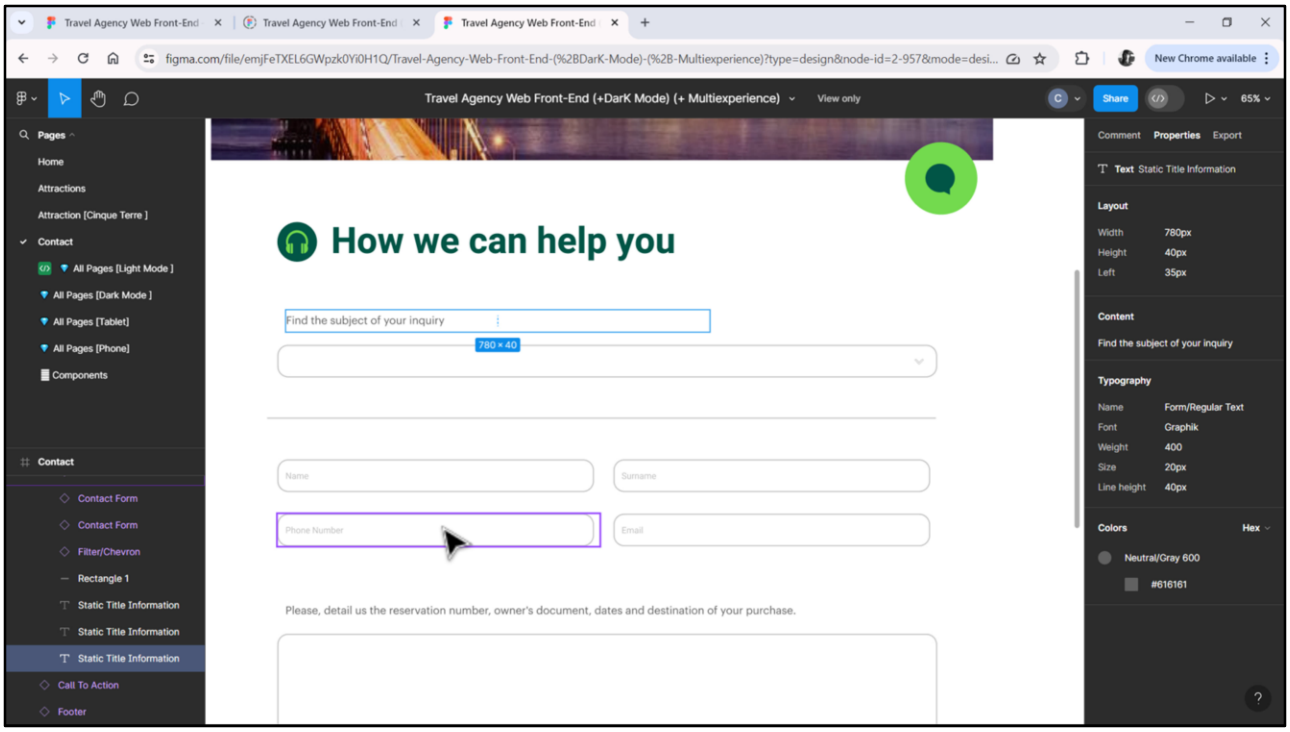


Mas agora o que me interessa é como modelamos o texto que aparecerá acima?

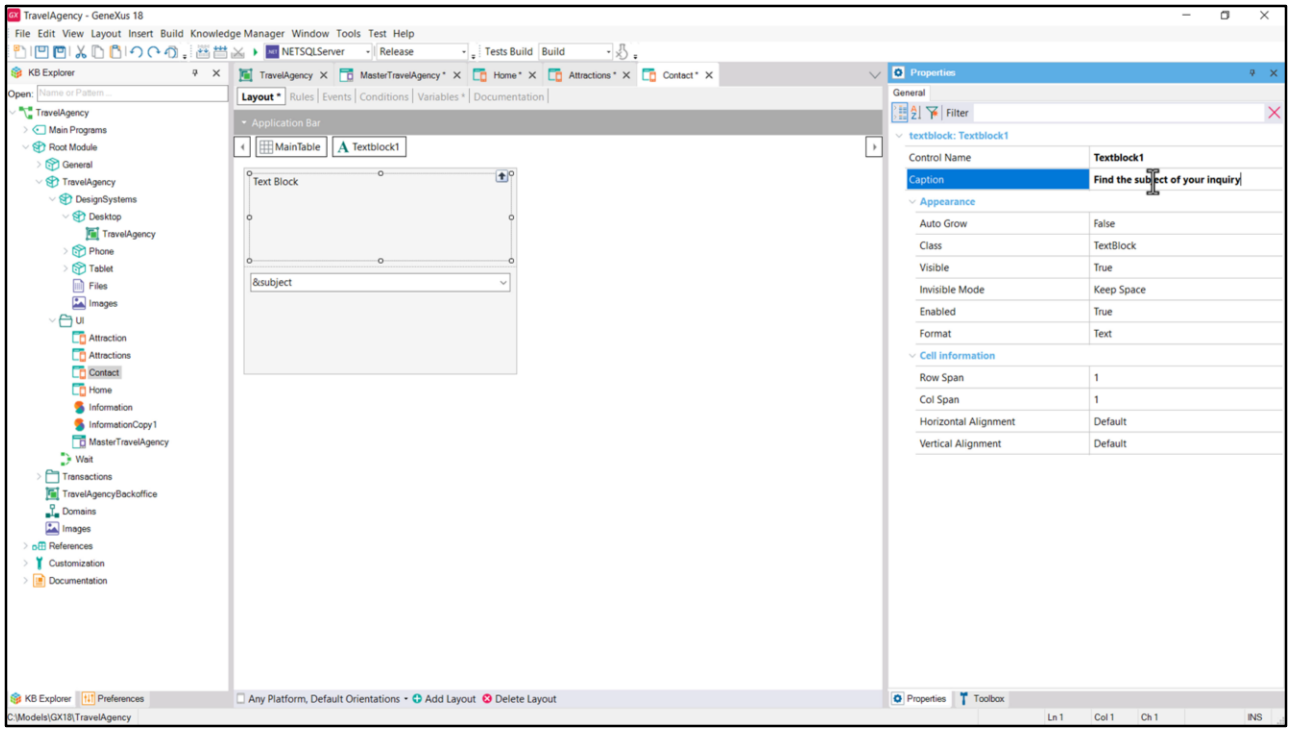
A forma nativa e mais óbvia é utilizar a propriedade Label Caption da variável e então colocar esse rótulo acima.
O valor que assumirá por padrão a propriedade Accessible Name Custom será este mesmo…



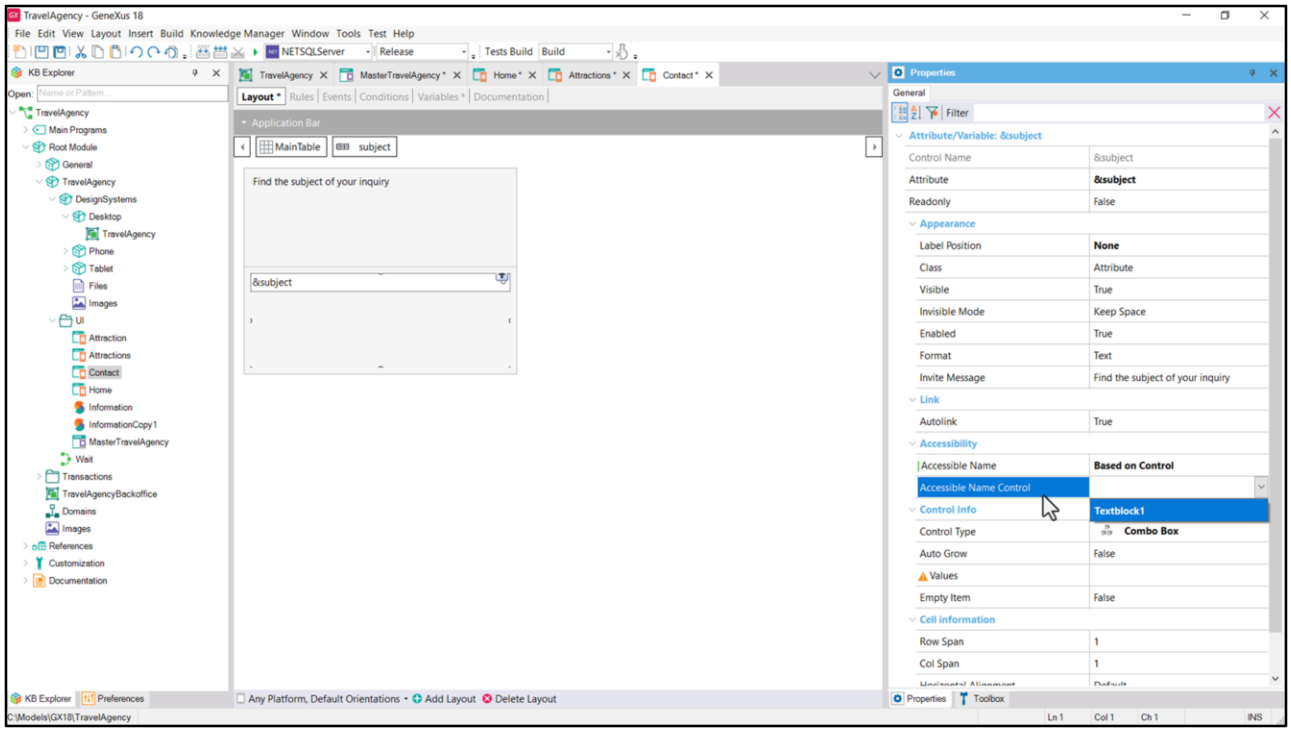
Observemos, por outro lado, que se não quisermos que o rótulo fique visível, tendo inserido este valor, ele será o que assumirá por padrão a propriedade Accessible Name Custom.



Isso será útil para estas outras variáveis que não têm texto descritivo visível.

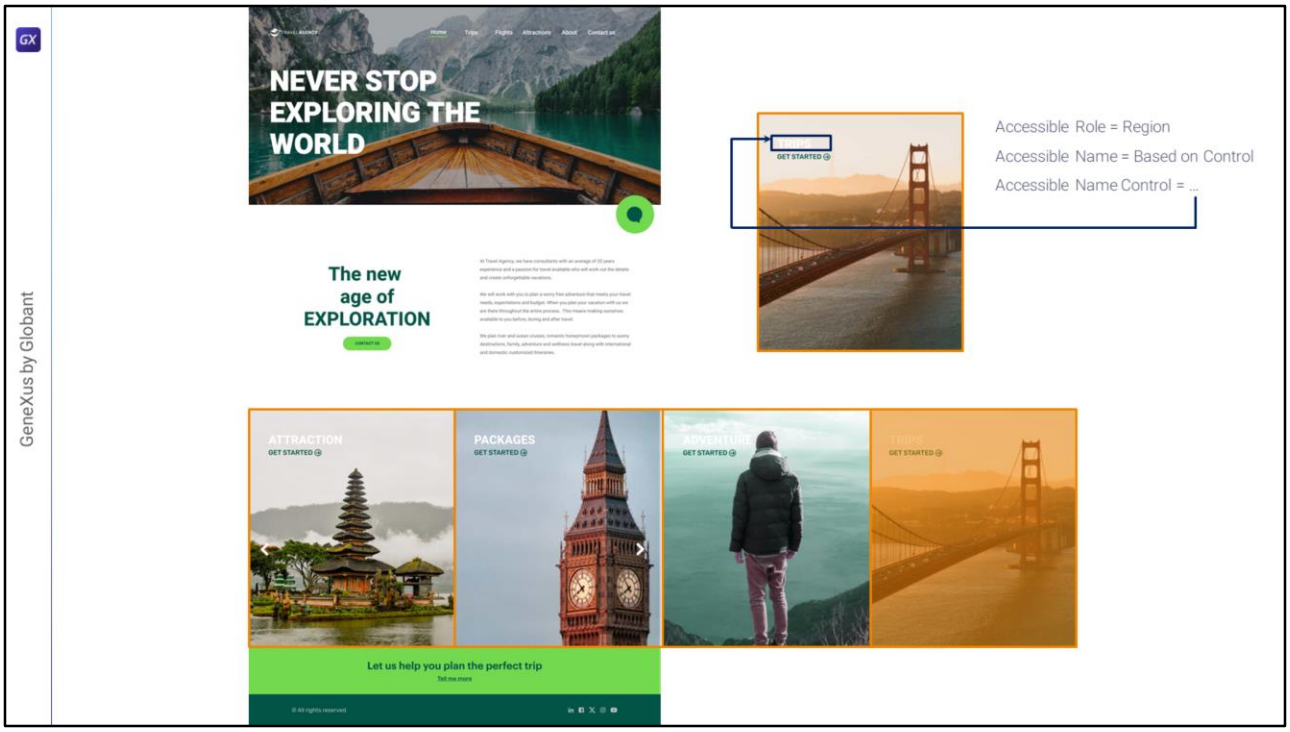


Mas também poderíamos ter pensado em implementar o label de maneira independente, ou seja, por meio de um controle Text block (opção que desaconselhamos). Então inseriríamos o controle, chamado Textblock1, especificaríamos seu caption…



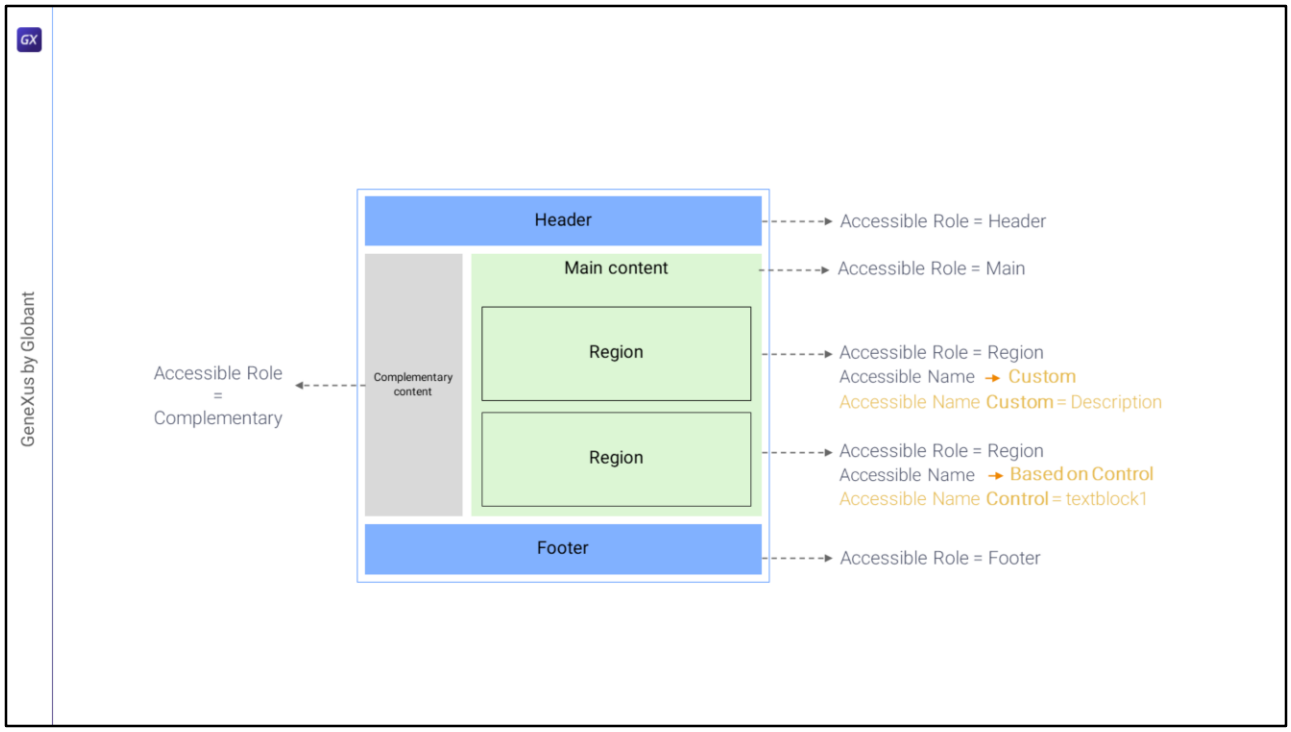
…e o que teríamos que fazer é dizer ao nosso combo box para pegar seu nome para a acessibilidade desse controle.

Mas ao fazer isso, ao clicar no rótulo, o label, não será feito foco na variável, como aconteceria com a primeira solução, pois neste caso se trata de dois controles independentes, enquanto no primeiro caso o rótulo fazia parte do campo.



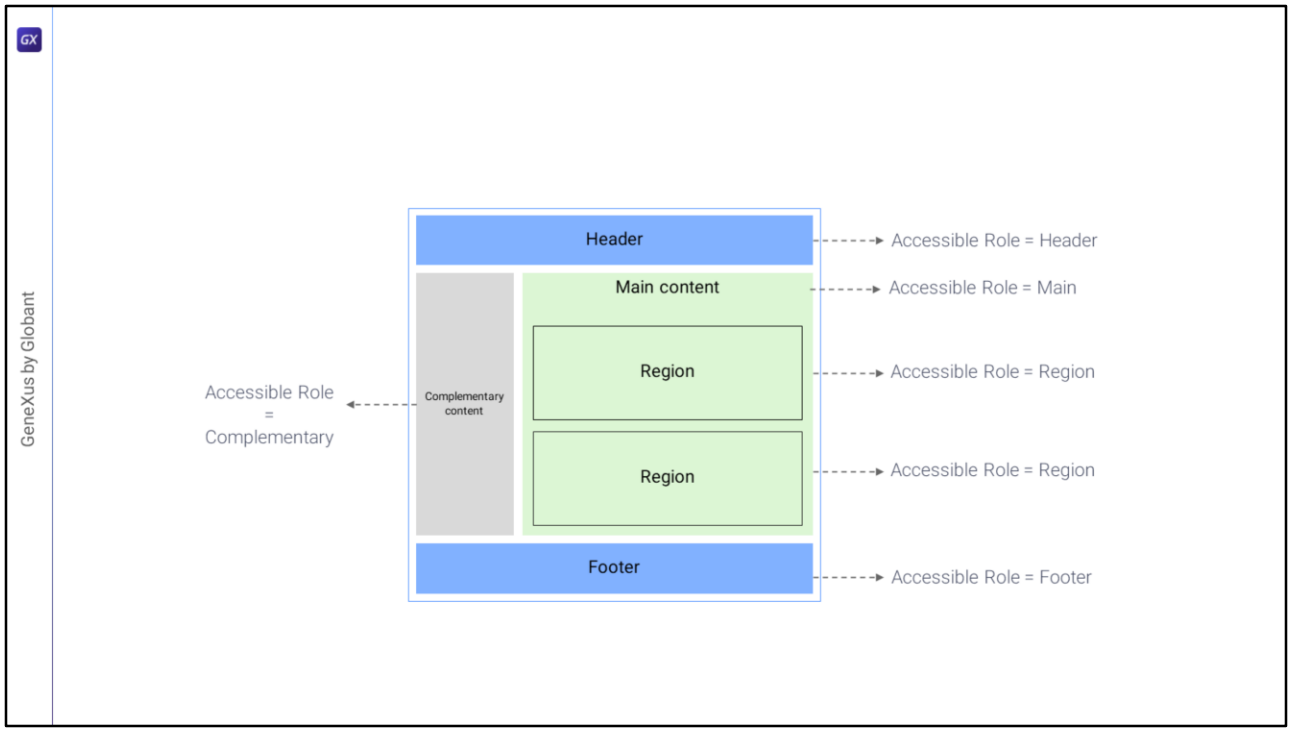
A propriedade Based on control aparece para aqueles casos em que o controle não possui rótulo de forma nativa entre suas propriedades e ainda assim há um text block no layout que pode descrevê-lo.

Por exemplo, o carrossel de Home conterá 4 cards (2 visíveis). Implementaremos isso como um grid horizontal onde cada item será um card. E cada um desses cards será implementado através um container canvas (que é como uma tabela que permite sobrepor controles, como veremos). Então, para cada um desses canvas, colocaremos a propriedade Accessible Role Region, mas o Accessible Name, observemos que já o temos no próprio layout. É o que corresponde ao título de cada Card, que será implementado como um text block. Então, em Accessible Name colocaremos Based on Control e em Accessible Name Control colocaremos o nome deste text block.



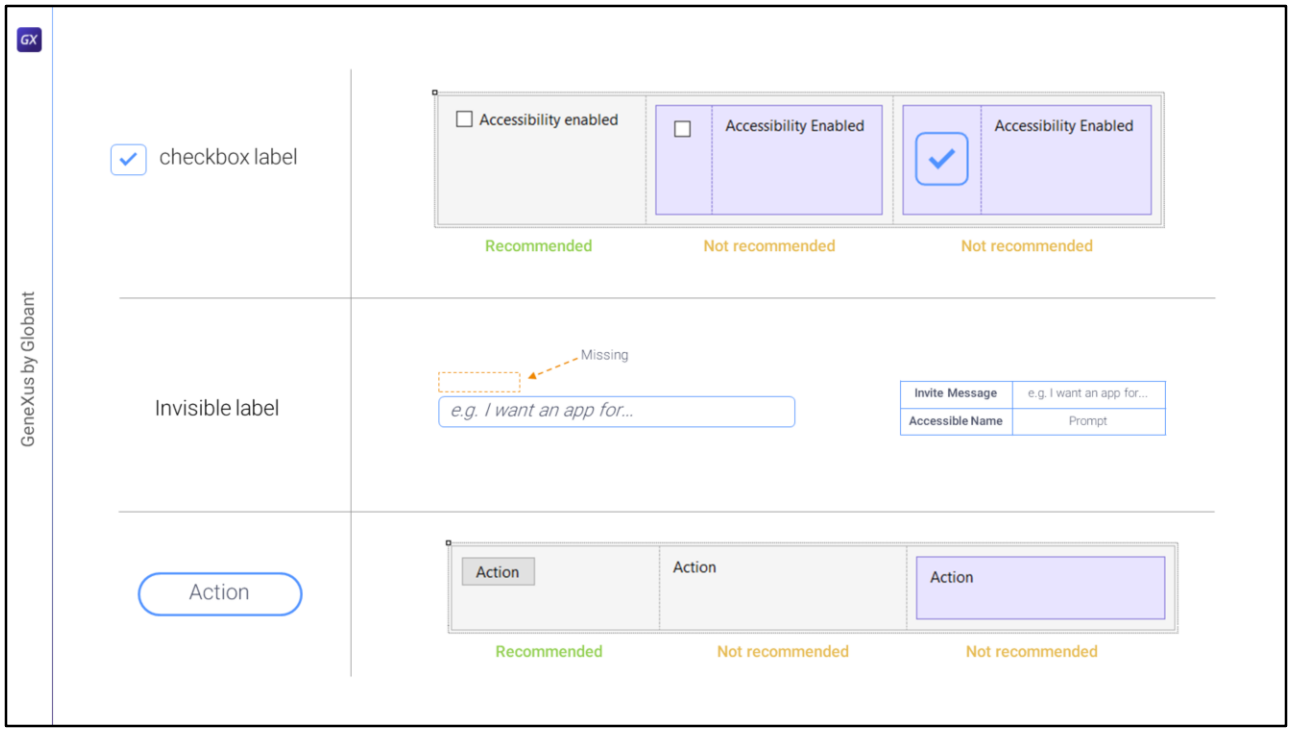
Em resumo, para os containers (e somente para eles) temos que indicar a função que cada um desempenha na tela.

Quando se trata de uma função que pode ser repetida, como a das regiões, então damos a descrição semântica à região por meio da propriedade: Accessible Name Custom se vamos fornecer diretamente o valor ali mesmo de forma customizada, escrevendo-o diretamente... ou através da propriedade Accessible Name Control se quisermos que pegue seu valor de um controle text block que esteja por ali.



A propriedade Role no momento, só terá efeito para a aplicação Angular e não para Android ou Apple. E só se aplica aos containers.

Mas as outras duas propriedades também são válidas para outros controles como os botões, as imagens, os atributos/variáveis, como vimos. Também para os grids. Estas também valem para Android e, neste momento, estavam terminando de implementar para Apple (se é que já não terminaram).

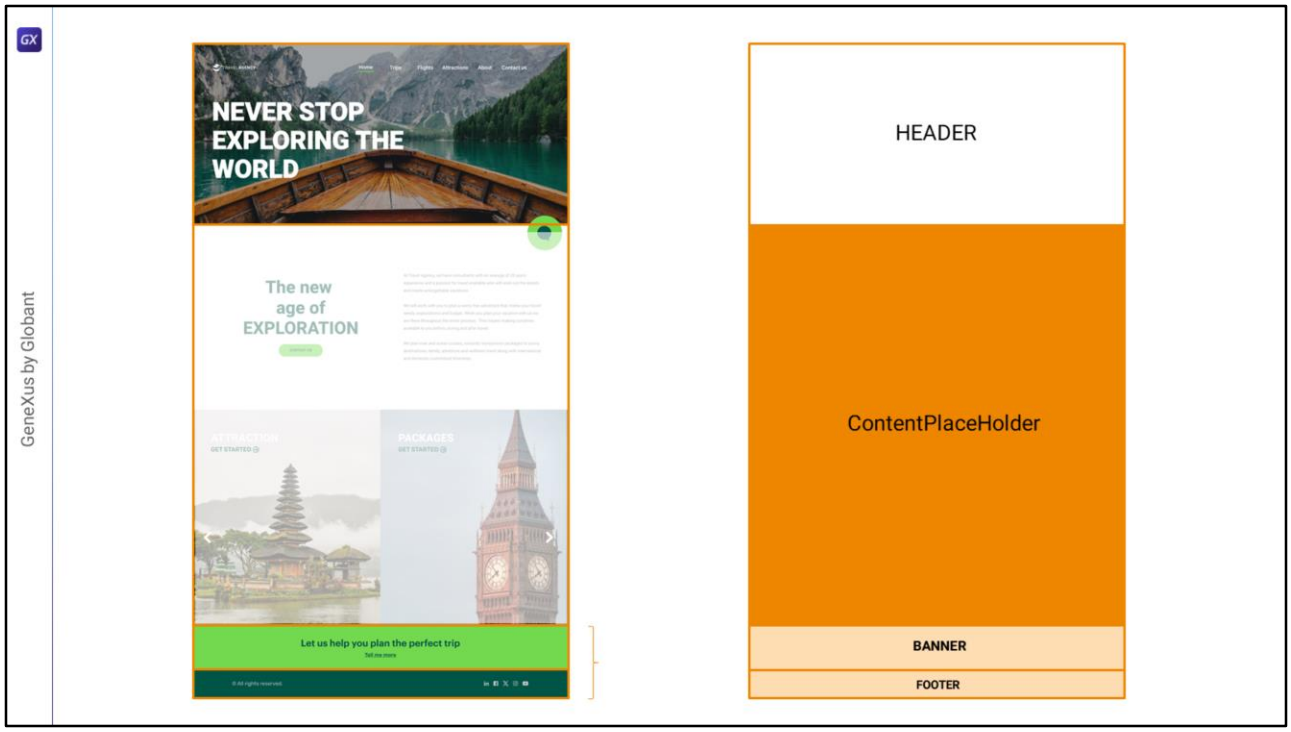


Bem, então vimos como recomendações para a acessibilidade: dividir a tela em containers com funções especificadas e depois, para os demais controles:

- Utilizar o rótulo(label) do próprio controle e não um controle text block separado. Por exemplo, a maneira recomendada de implementar um checkbox é através de um controle attribute/variable com tipo de controle checkbox e com a indicação semântica através de seu próprio Label Caption, e não com um text block separado para isso, ou mesmo utilizando um controle imagem e um text block em vez do controle nativo, checkbox.

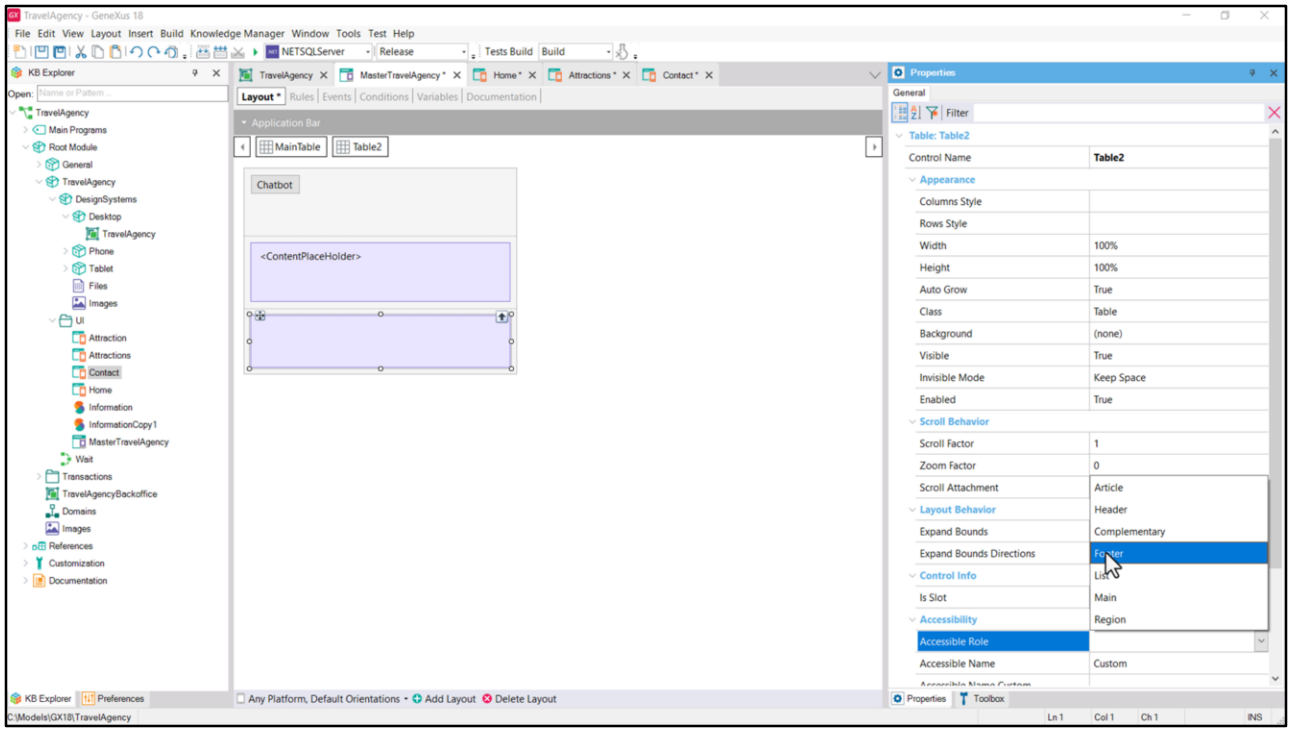
- Se o controle não tiver rótulo ou não será visível, não se esqueça de dar valor à propriedade Accessible Name Custom. Neste exemplo temos uma variável que utiliza uma mensagem interna no campo que, quando o usuário começa a digitar nele, desaparece. Não é o texto descritivo, é uma mensagem de convite para começar a inserir valor ali. Esta mensagem é fornecida através da propriedade Invite Message do controle, mas não tem nenhum impacto na acessibilidade. Precisamos nos preocupar que o campo tenha atribuído valor semântico na propriedade Accessible Name.

- E por último, utilizar botões sempre que desejar implementar uma ação e não outro tipo de controle com evento associado, como um text block, um text block dentro de uma tabela ou mesmo (embora não seja mostrado aqui) uma imagem com evento associado. Em qualquer um desses casos se perde a semântica do botão, que é justamente implementar uma ação. Para ações devem ser sempre utilizados botões, aos quais depois é dada a estética desejada, que pode ser sem bordas, sem moldura, somente texto, ou somente imagem, ou ambos, mas onde seu “ser botão” esteja garantido.



Bem, tudo isso colocaremos em prática nos vídeos seguintes, à medida que vamos implementando os layouts.

Então, voltando à implementação do Master Panel. Na linha um teremos que implementar o Header, e na 3 e 4 essas outras partes, que Chechu chamou em Figma de Banner e Footer. No entanto, Banner no nível da semântica web é entendido como Header, e na realidade essas duas seções funcionam melhor como um Footer…



Então deixaremos uma única linha na qual colocaremos uma tabela com a função Footer.

A implementação desta parte é mais simples, então vamos mergulhar na mais complexa, que é a do Header. Vamos dedicar-nos a ela no que vem a seguir.

Continuamos no próximo vídeo.


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