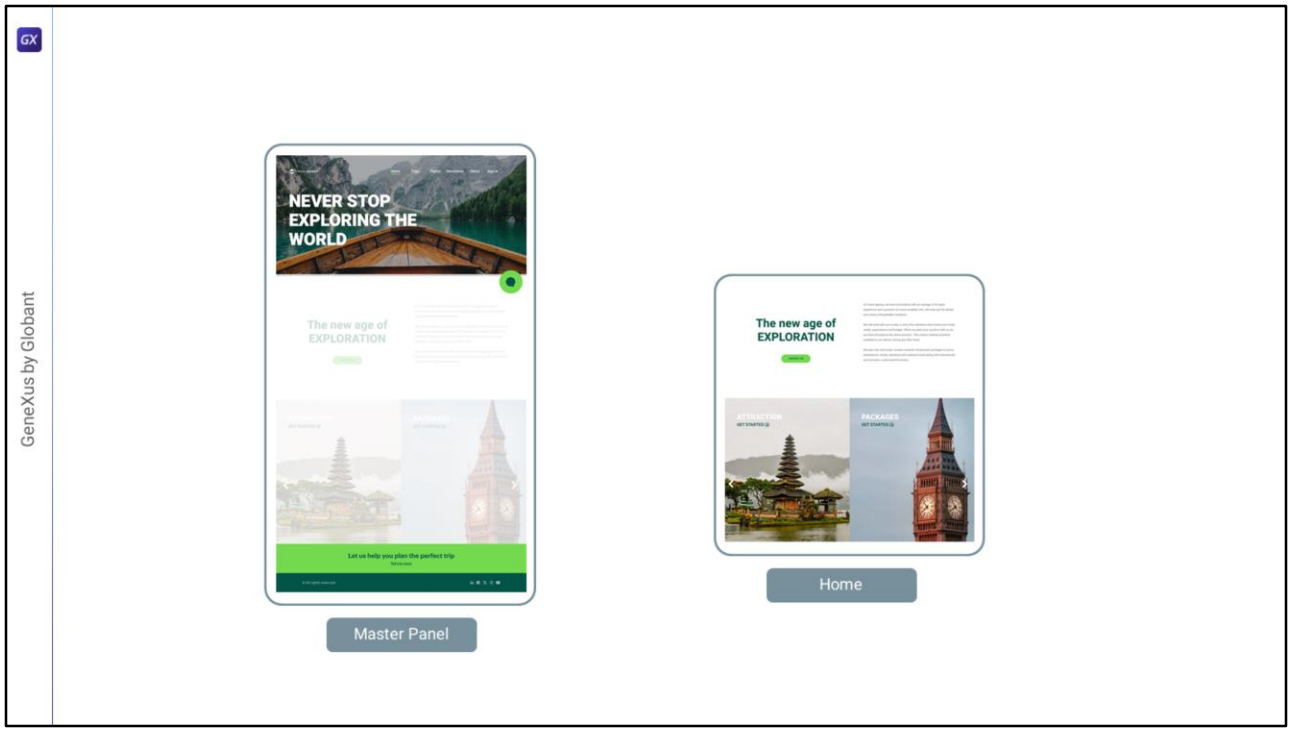
GeneXus by Globant

Master Panel:
Header update and Navigations

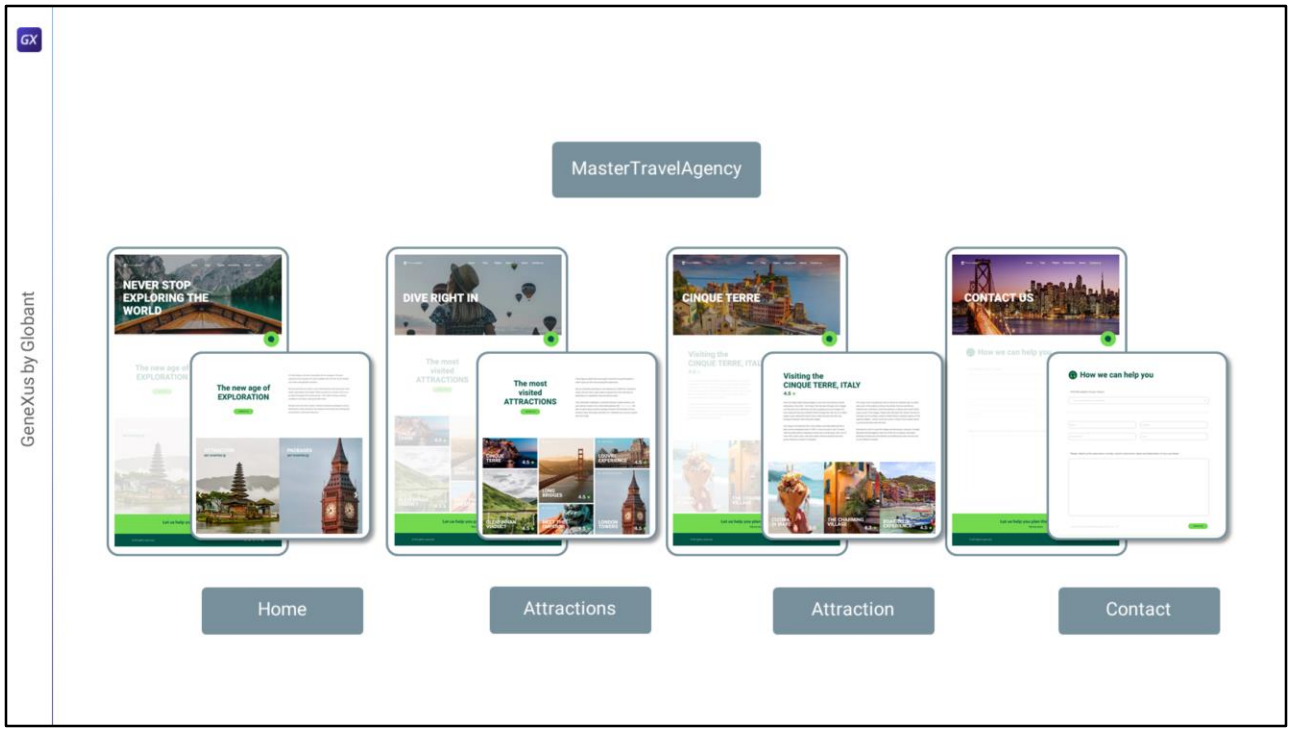
Cecilia Fernández



Bom, neste vídeo vamos completar a implementação do Header.

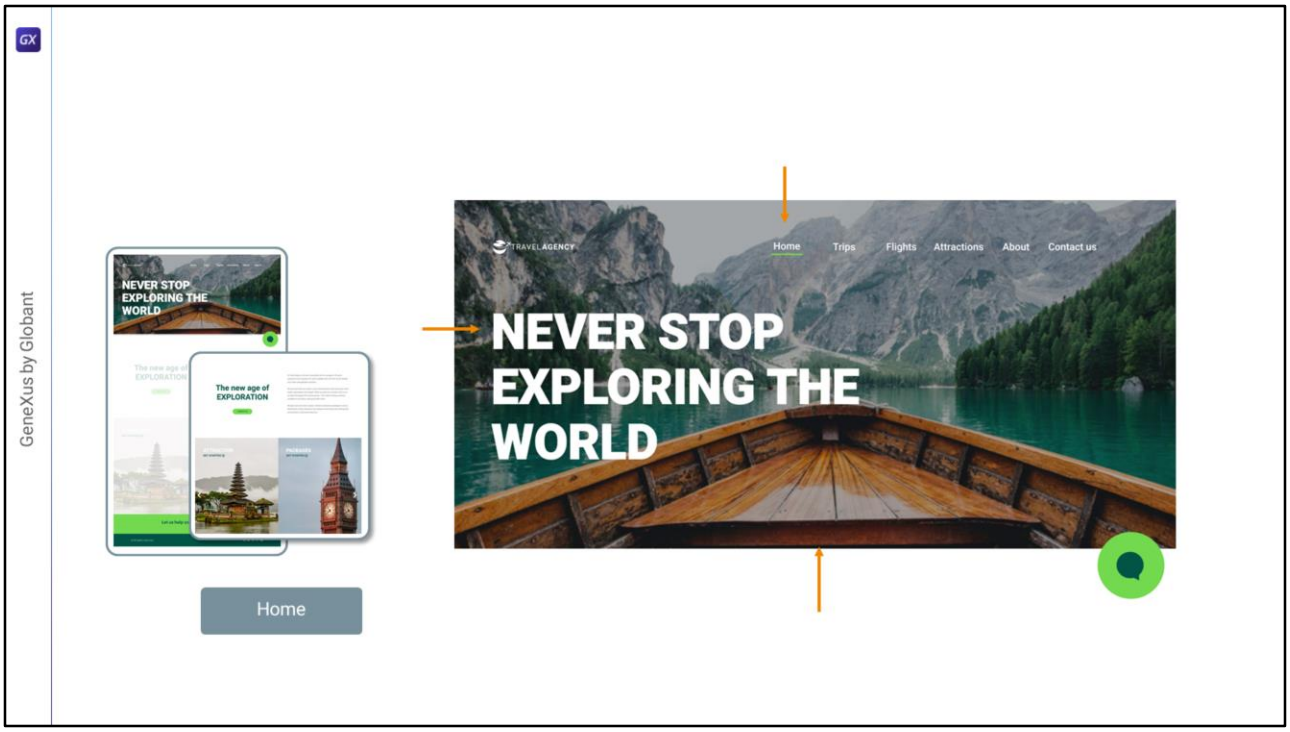
Como sabemos, o Master Panel não é um objeto executável de forma independente. O Master Panel é executado toda vez que um panel que o tem como Master Panel é executado.

Como já sabemos, o layout do panel será renderizado dentro do controle ContentPlaceHolder do Master Panel.

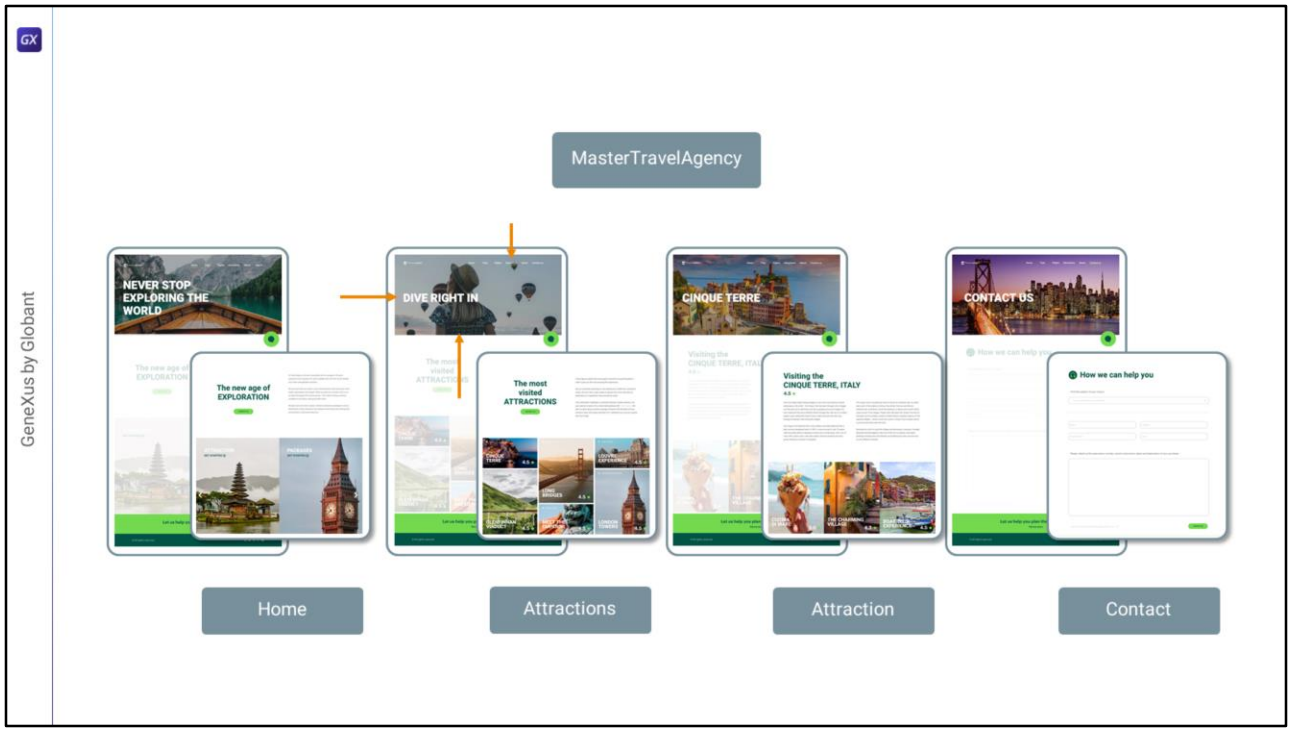


No nosso caso, esses quatro painéis serão os executáveis. Mas, cada vez que um deles for invocado, executará o Master Panel.

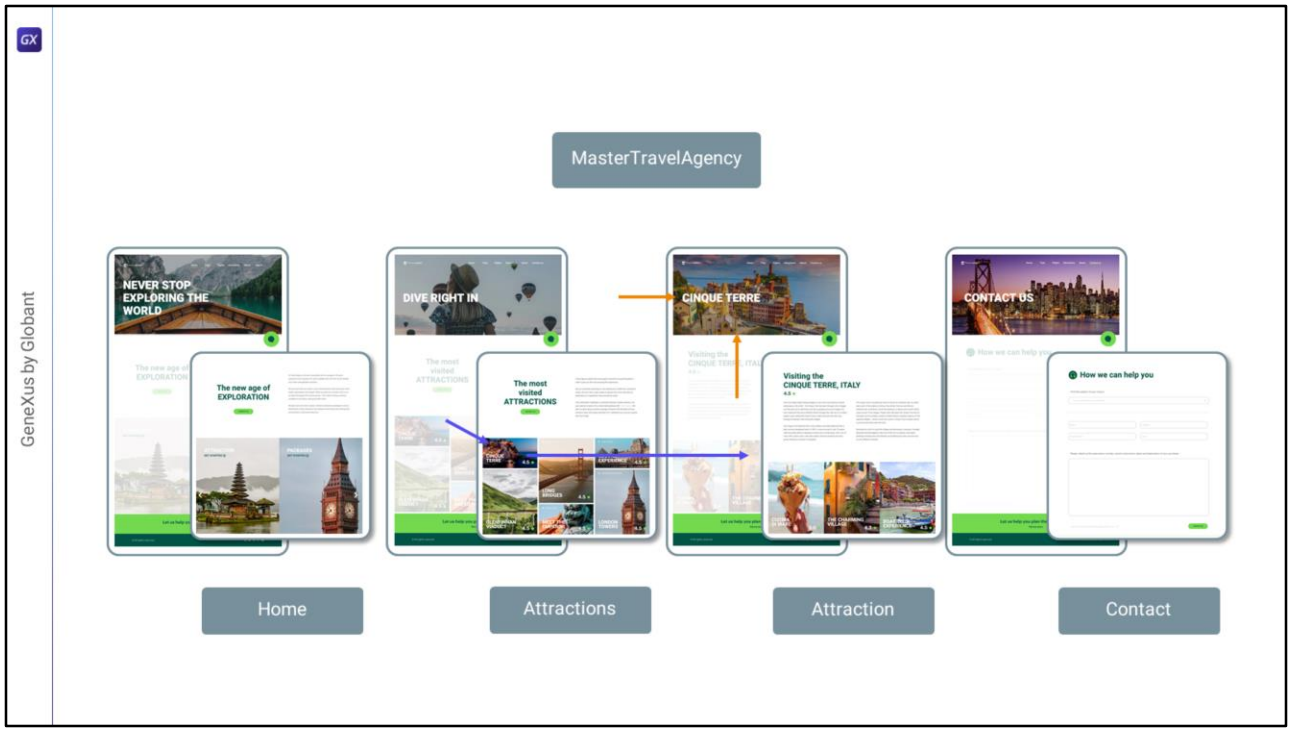
No nosso caso, o Master Panel deverá saber quem o mandou executar porque esse “quem” determina três coisas: a imagem de fundo do Header, o título sobre a imagem e a opção do menu que deve ficar como selecionada.



Para o Home, deverá ser exibida esta imagem, este título e o botão que deverá ficar como selecionado (com o indicador verde abaixo) é o Home.

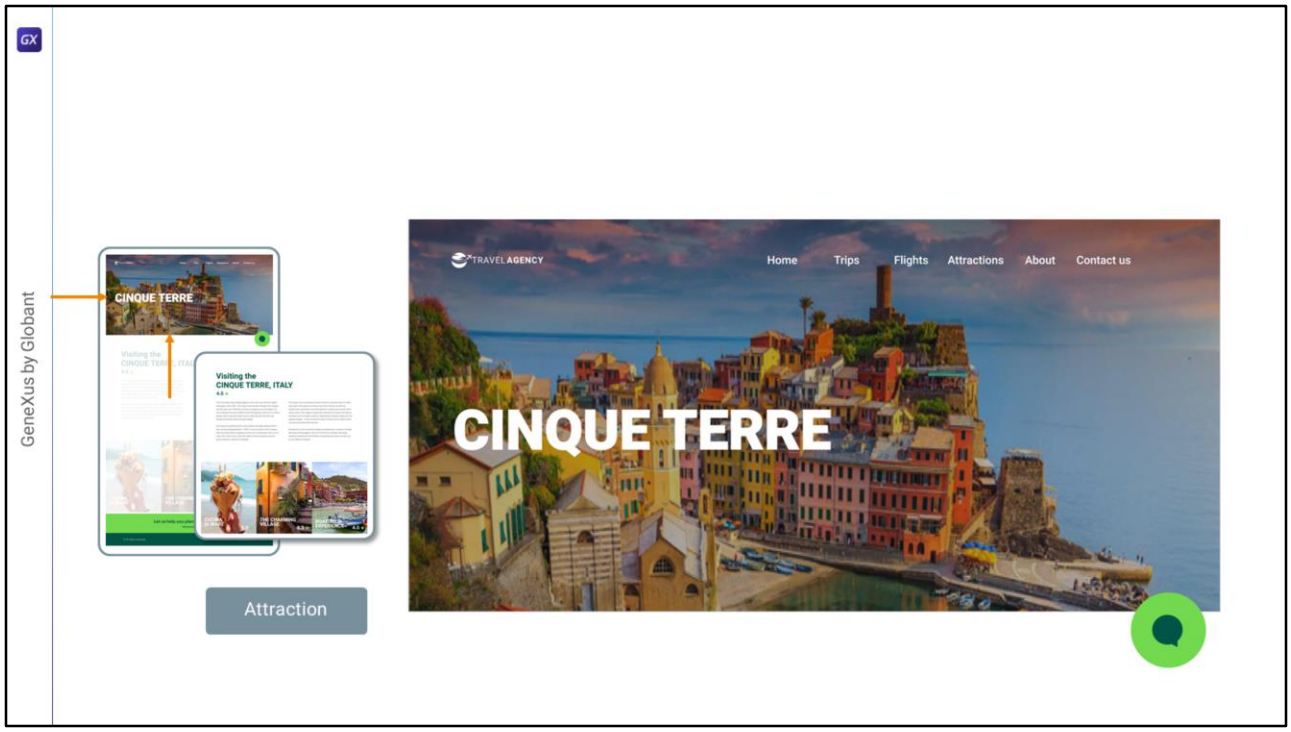


Para Attractions, estas deverão ser a imagem e o título, e o botão selecionado deverá ser Attractions.

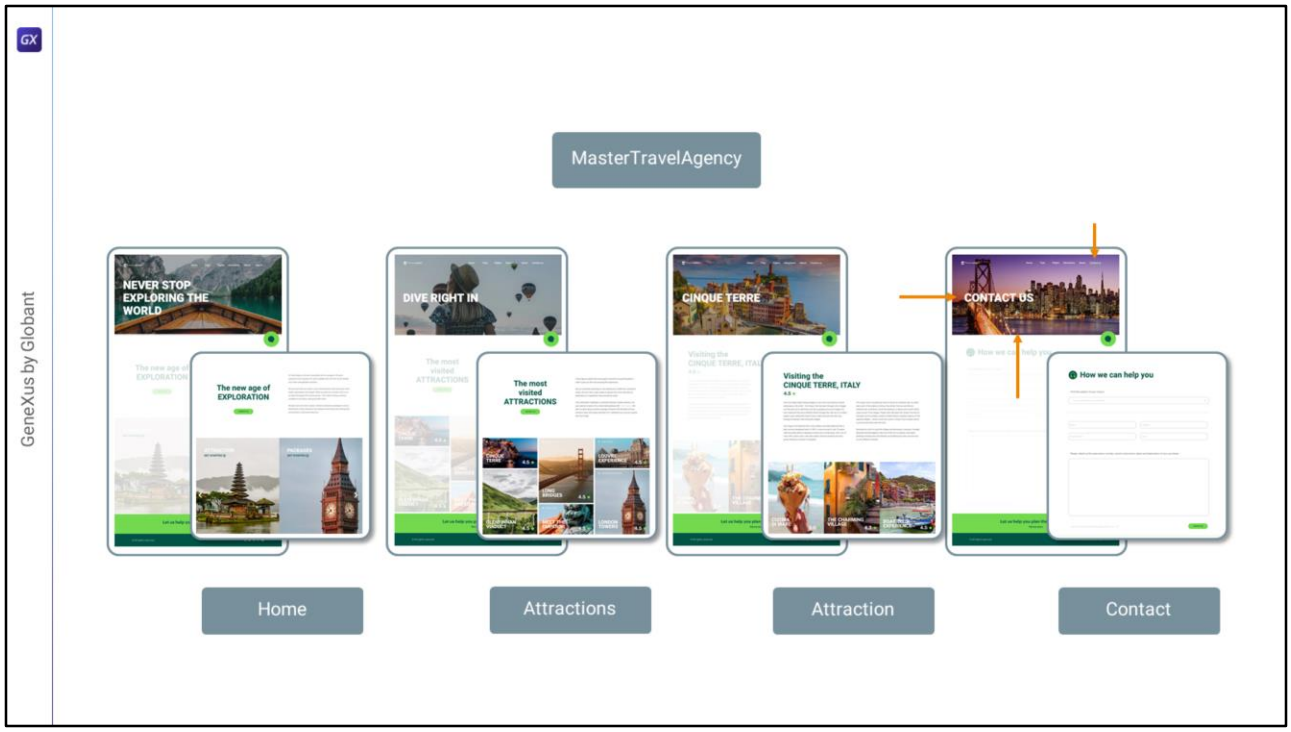


Para Attraction acontece algo diferente, porque não é uma opção do menu. Esta página será acessada a partir de outra, escolhendo alguma das atrações turísticas exibidas neste carrossel. Então, este panel receberá o identificador de atração turística como parâmetro e deverá acessar a base de dados para carregar aqui a informação dessa atração turística.

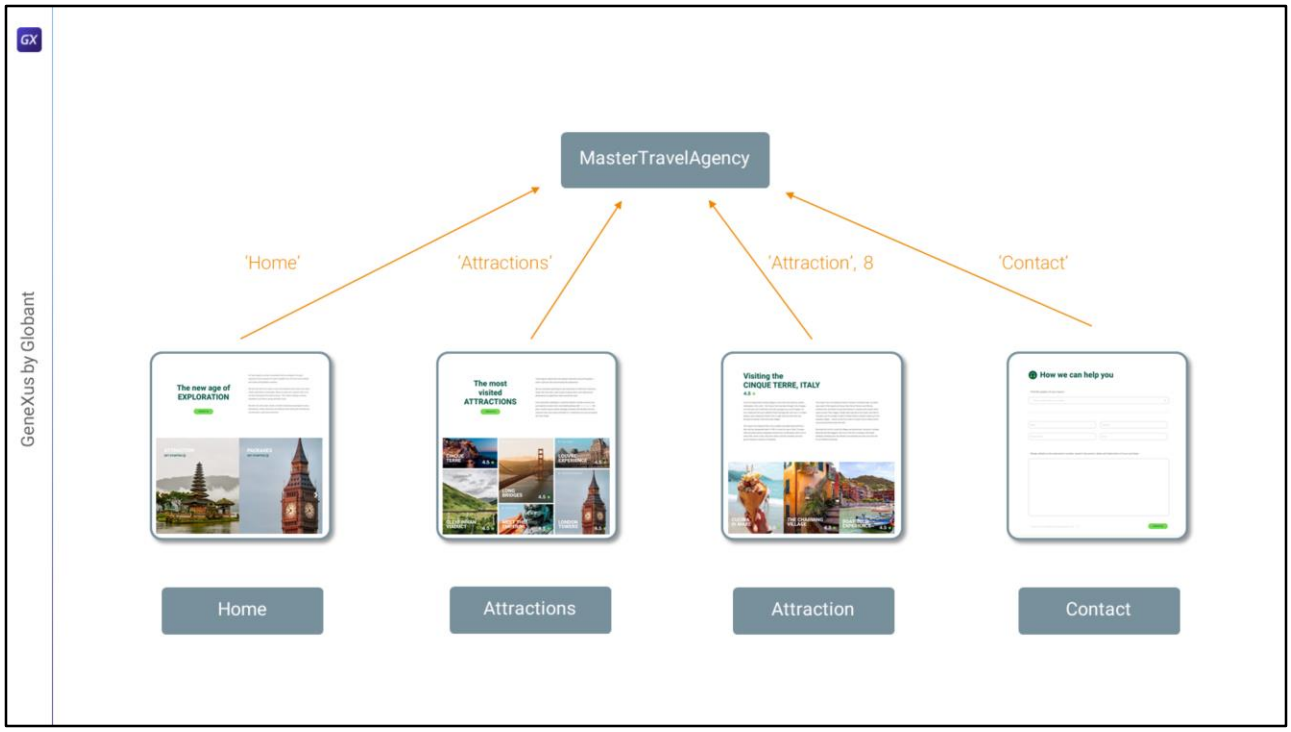
Mas, além disso, no nível do Master Panel, também deverá saber qual é essa atração turística, porque também terá que ir à base de dados para trazer sua foto, que é a que exibirá como Hero no Header, e seu nome para exibir como título…



...e, no caso do menu, na verdade, deverão aparecer todas as opções desmarcadas.



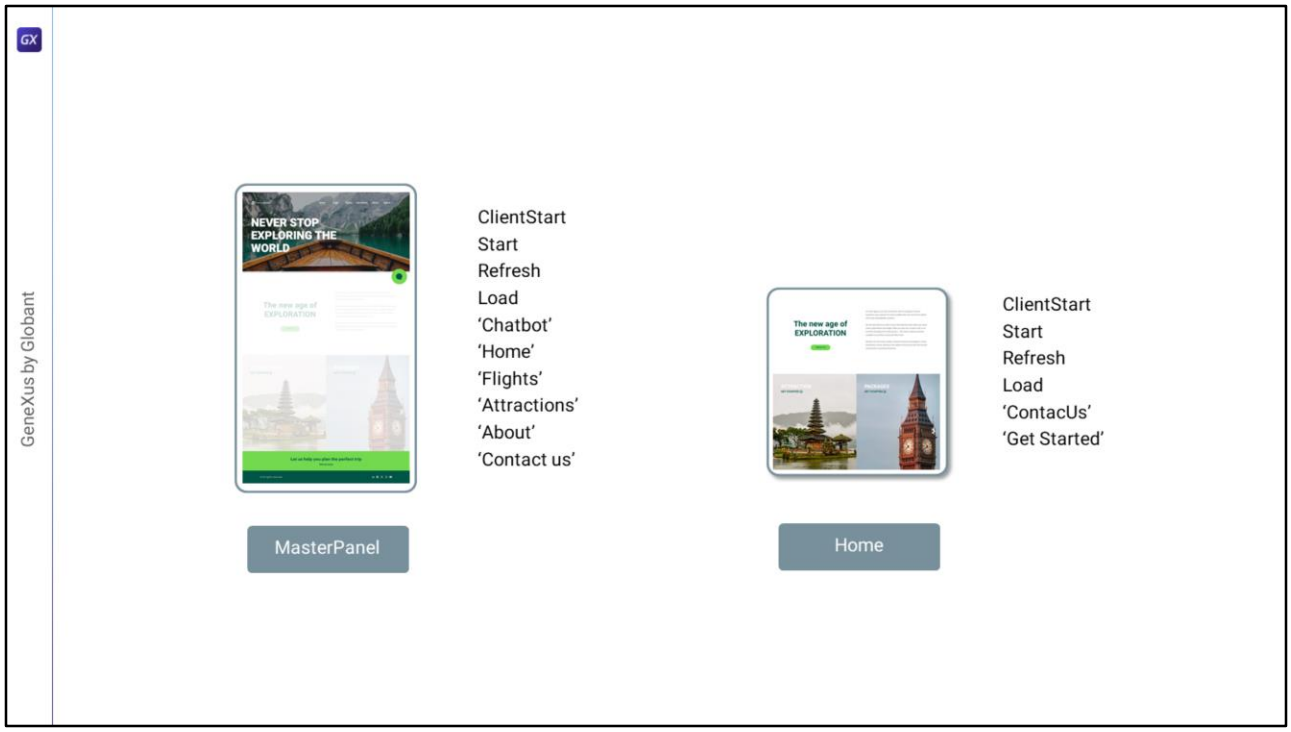
O panel do contato é igual aos dois primeiros, com imagem, texto e botão selecionado fixos.



Em resumo, o panel que estiver sendo executado em cada oportunidade deve se identificar perante o Master Panel, para que este possa tomar as ações necessárias sobre os 3 elementos do Header que acabamos de analisar: imagem de fundo, título e botão selecionado.

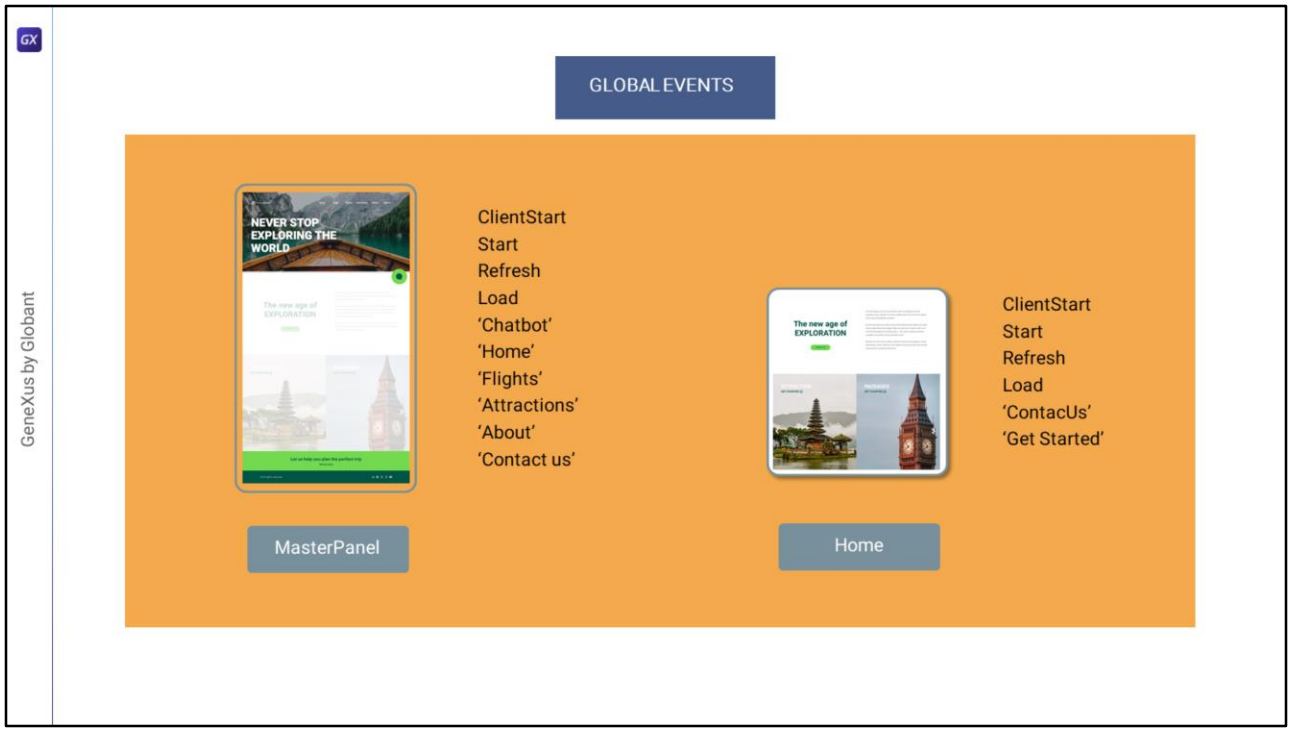
Poderíamos pensar que isso se resolve com passagem de parâmetros, ou seja, cada panel envia por parâmetro ao Master Panel uma identificação de quem é. Mas o Master Panel não admite passagem de parâmetros.

Então? Como podemos fazer para produzir essa comunicação entre objetos que não podem passar parâmetros? Uma boa opção é utilizar os eventos globais.

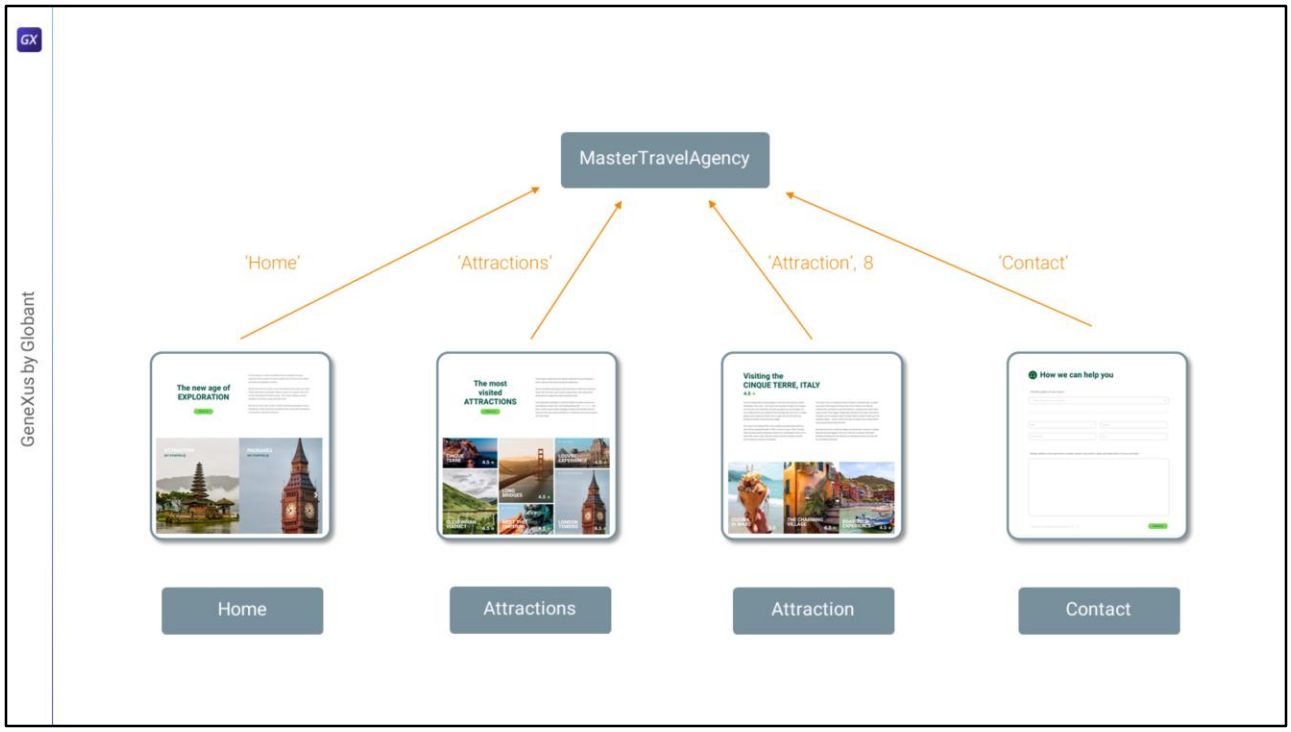


Os eventos com os quais estamos familiarizados até o momento são os eventos de cada objeto, internos ao objeto, tanto disparados pelo usuário quanto pelo sistema. Por exemplo, o ClientStart, o Start, o Refresh, o Load, que são do sistema, ou os eventos associados aos controles da User Interface.

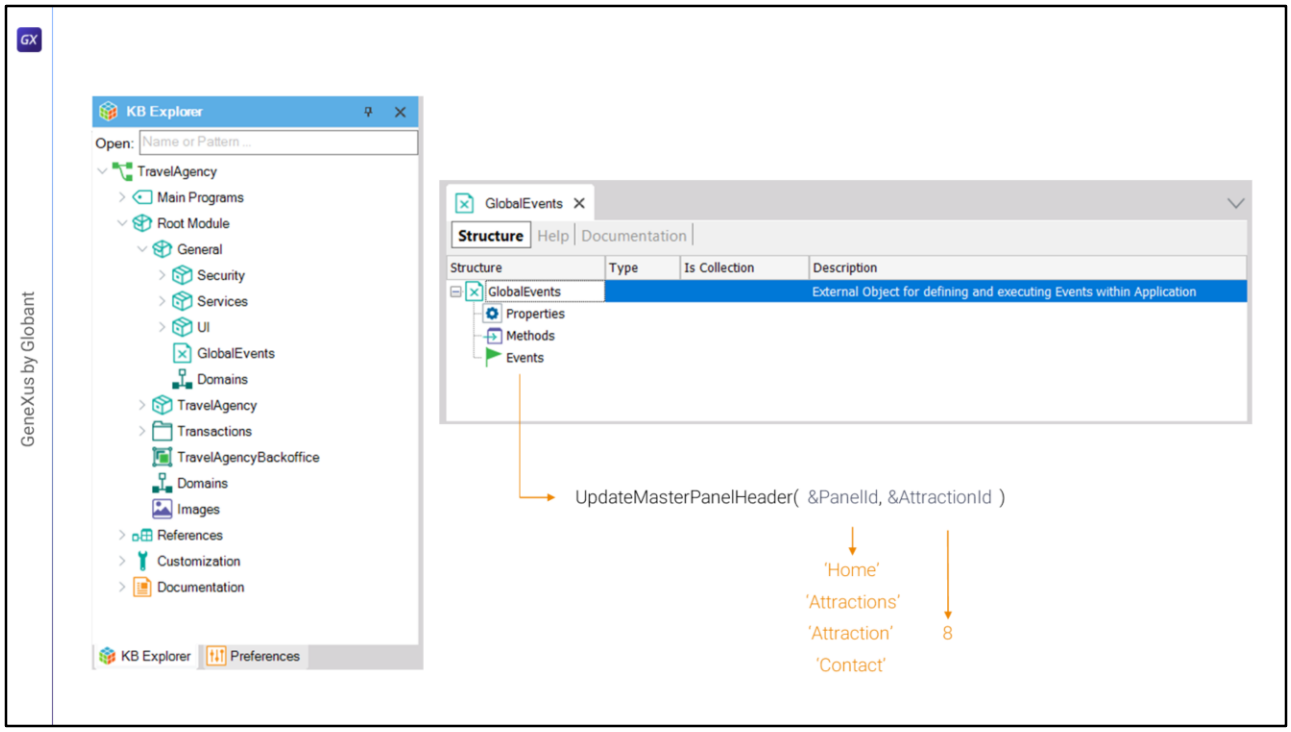
Mas esses são todos eventos locais, são disparados e tratados dentro do objeto que os define, internamente.



Em vez disso, os **eventos globais** serão válidos para todos os componentes do contexto de um objeto que está sendo executado. No nosso caso, o Panel e o Master Panel estão dentro de um mesmo contexto de execução. Portanto, se o Panel disparar um evento global, o Master Panel, que está no mesmo contexto de execução, poderá escutá-lo e executá-lo. O mesmo será válido para componentes, como eu vou contar a vocês mais adiante.



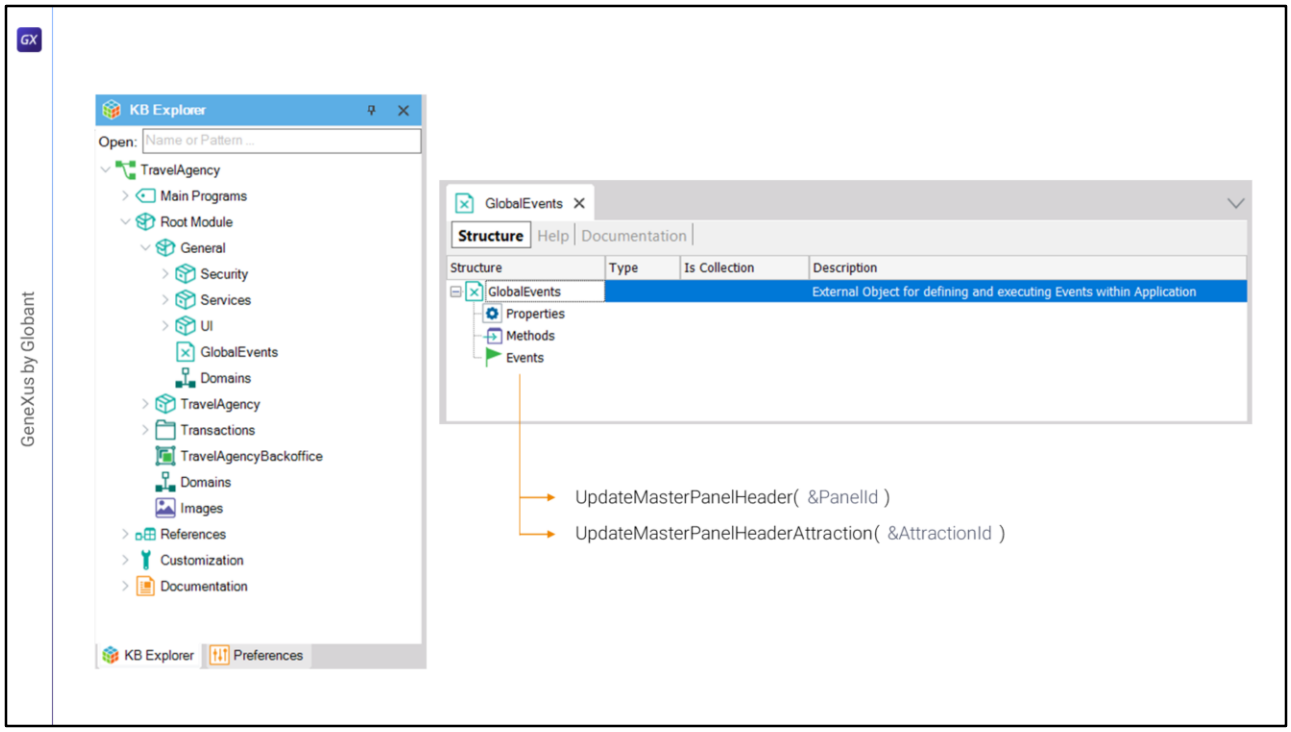
E isso é o que precisamos. Que os painéis disparem um evento que seja escutado e executado pelo Master Panel, através do qual possam passar informação para o Master Panel.



Para isso é que toda KB disponibiliza dentro do módulo General um objeto GlobalEvents, válido para todas as plataformas (web e nativas).

Lá, sob o seu nó Events, é onde poderemos definir eventos globais, para que possam ser compartilhados entre objetos em um mesmo contexto de execução.

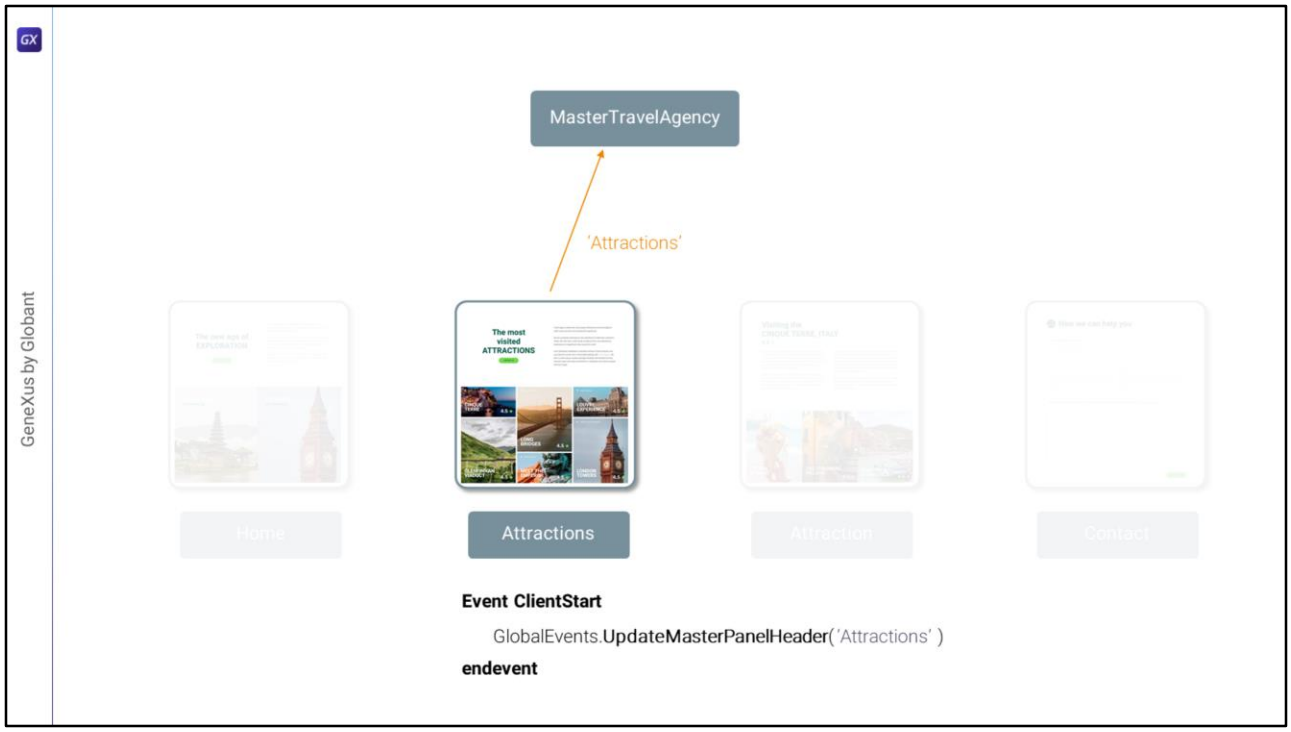
Para o nosso caso, poderíamos definir um único evento, que ao ser disparado (no nosso caso a partir dos 4 panels) deverá enviar a quem escutar (no nosso caso o Master Panel) **2 parâmetros**: o primeiro para identificar o objeto que o está disparando (será 'Home', 'Attractions', 'Attraction' ou 'Contact'), e o segundo só terá sentido se quem o estiver disparando for Attraction, pois é para identificar a atração turística.
Se quem o estiver disparando for Home, Attractions ou Contact, o valor passado para o segundo parâmetro não será considerado pelo Master Panel.



A outra opção é diferenciar dois eventos, um que será invocado a partir de Home, Attractions ou Contact para se identificar, e outro que será disparado unicamente a partir de Attraction, enviando o Id da atração.

As duas soluções (um único evento versus dois diferenciados) têm prós e contras. Vou escolher a segunda, com dois eventos, apenas para que vejam que podemos adicionar todos os eventos globais que desejarmos a este objeto GlobalEvents.

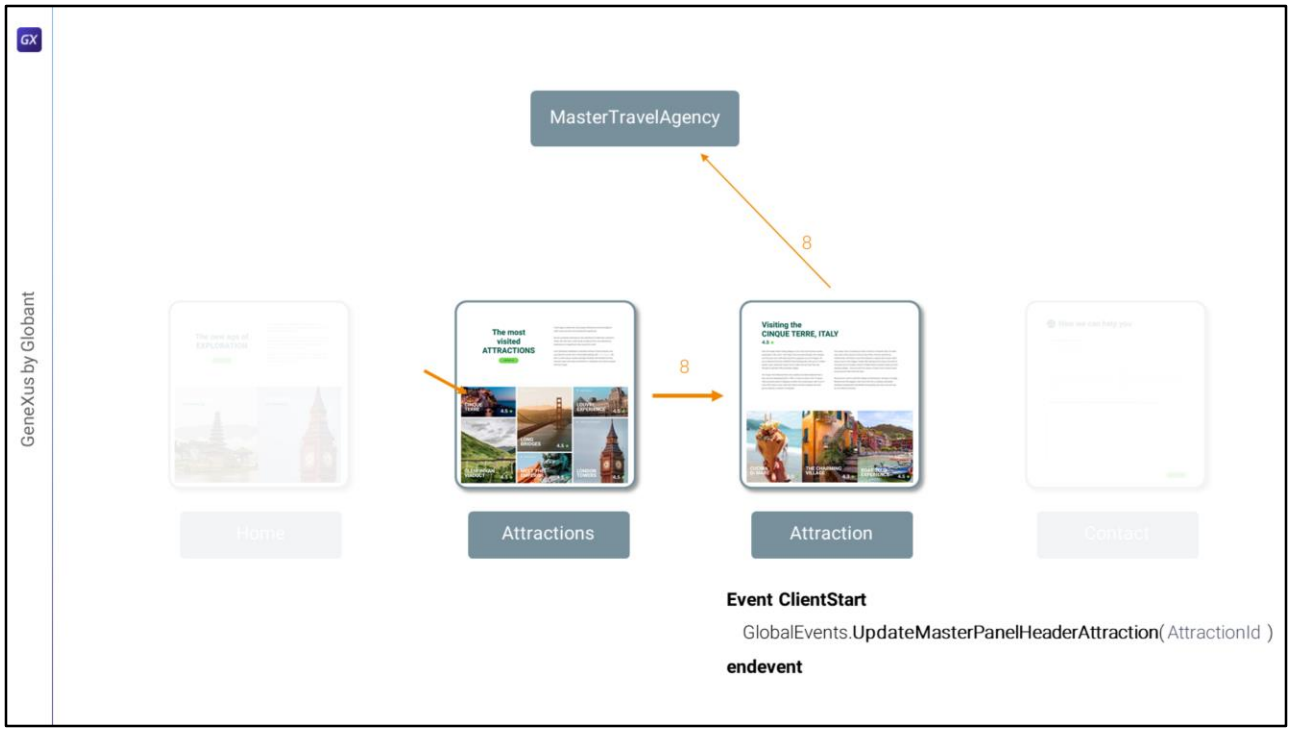
Então, para o panel Home, no ClientStart vou disparar o evento UpdateMasterPanelHeader, passando por parâmetro o valor 'Home'.



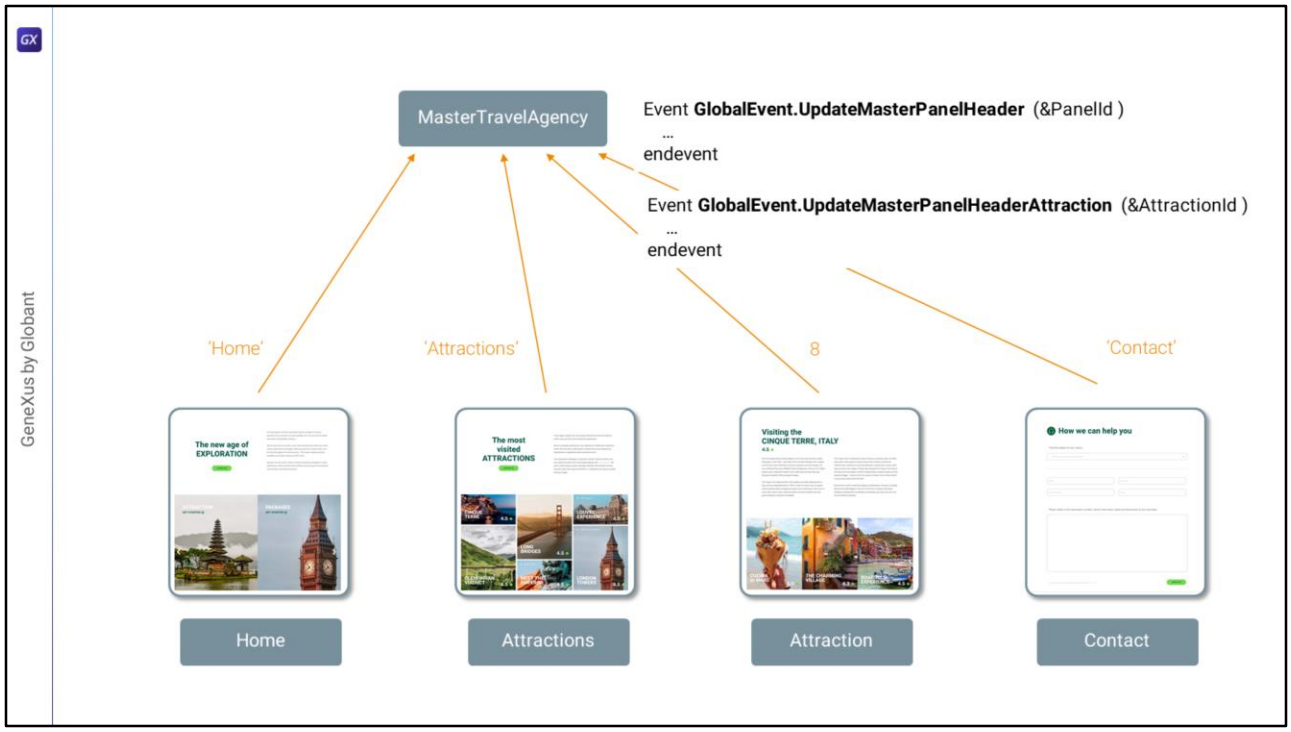
E o mesmo será feito no ClientStart de Attractions, mas passando, claro, ‘Attractions’ por parâmetro…

…e no de Contact, mas passando agora ‘Contact’.

No de Attraction será disparado o outro evento, o particular, UpdateMasterPanelHeaderAttraction…

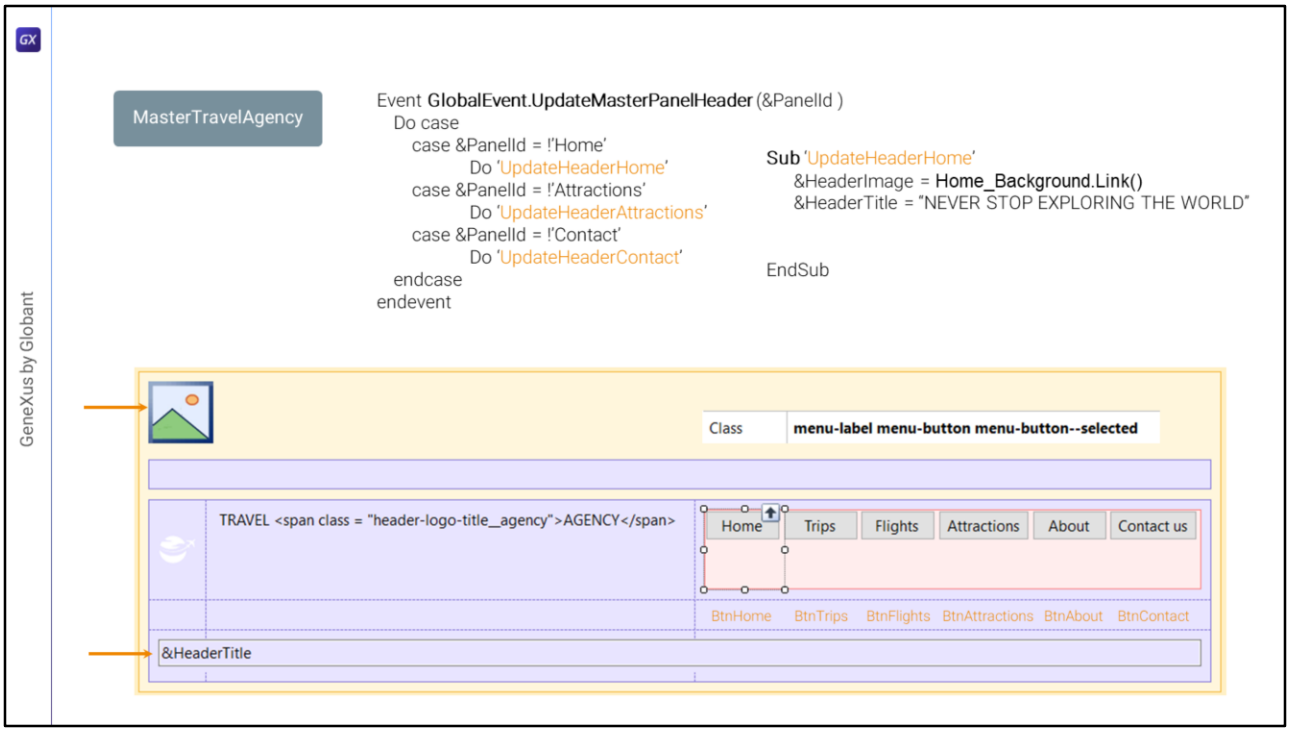


E como o panel Attraction receberá, por sua vez, por parâmetro o id da atração turística (que será enviado a partir daqui, quando o usuário selecionar uma atração para ver sua informação, por exemplo, a de id 8), esse mesmo id de atração será o que será enviado por parâmetro para este evento global.



Depois, o Master Panel só deverá “escutar“ esses dois eventos, para receber essa identificação e prosseguir.

Assim, se estiver sendo executado algum desses três painéis, dispararão este evento.

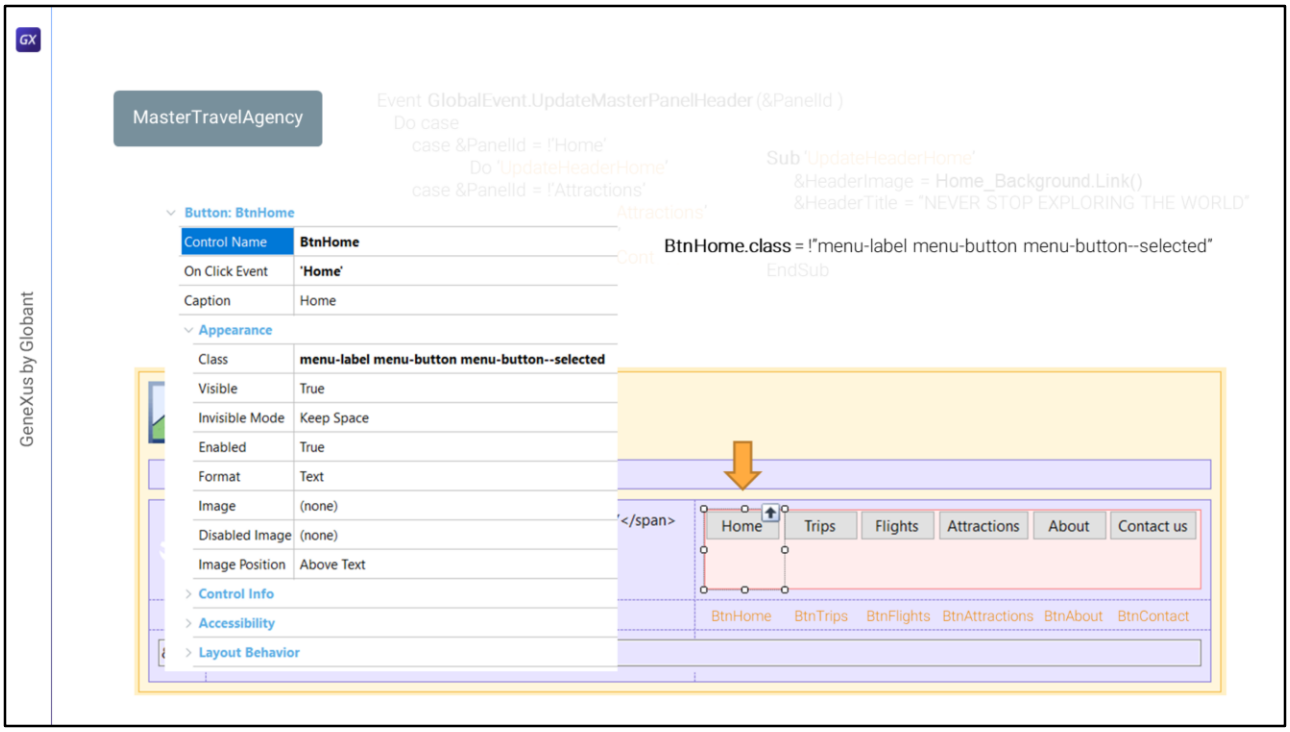


E o que o evento precisa fazer? Com base em quem o disparou... (este sinal de exclamação antes desses textos serve para indicar ao GeneXus que esses textos são internos à aplicação e, caso a aplicação seja traduzida para outros idiomas, esses textos não deverão ser traduzidos)... Bom, como dizia, com base em quem o disparou, deverá atualizar o Header carregando as duas variáveis: &HeaderImage e &HeaderTitle com os valores correspondentes para aquele panel que o disparou... e, para os botões, deverá deixá-los todos desselecionados, exceto um, o que corresponde ao panel.

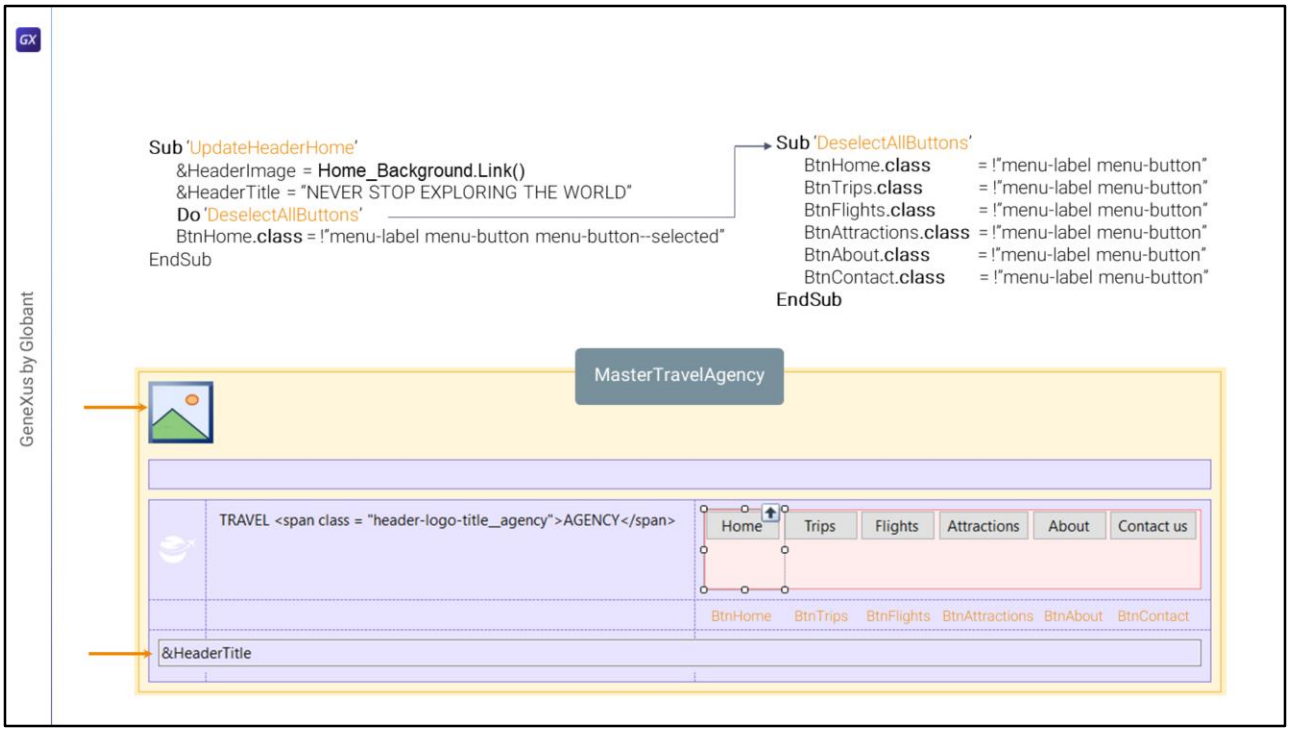
O que estamos escrevendo aqui são invocações a sub-rotinas, que são trechos de código locais ao objeto, que podem ser isoladas para reutilização ou para tornar mais compreensível a semântica do código.

Dentro de cada uma dessas sub-rotinas, por exemplo, vejamos a de Home... será atualizada, então, a variável da imagem, a do título... e o menu será deixado com a opção selecionada correta. Como fazemos este último?

Se nomearmos os controles botão desta maneira... lembremos do vídeo anterior que tínhamos duas classes para todos os botões: a menu-label para a tipografia e a menu-button para dar a borda inferior e superior transparente, lembram? (poderíamos ter unido as duas em uma só, mas, enfim, não o fizemos) e uma terceira classe para conseguir o indicador verde abaixo, que era a “menu-button--selected“, que no vídeo anterior havíamos deixado associada de forma estática ao botão Home.

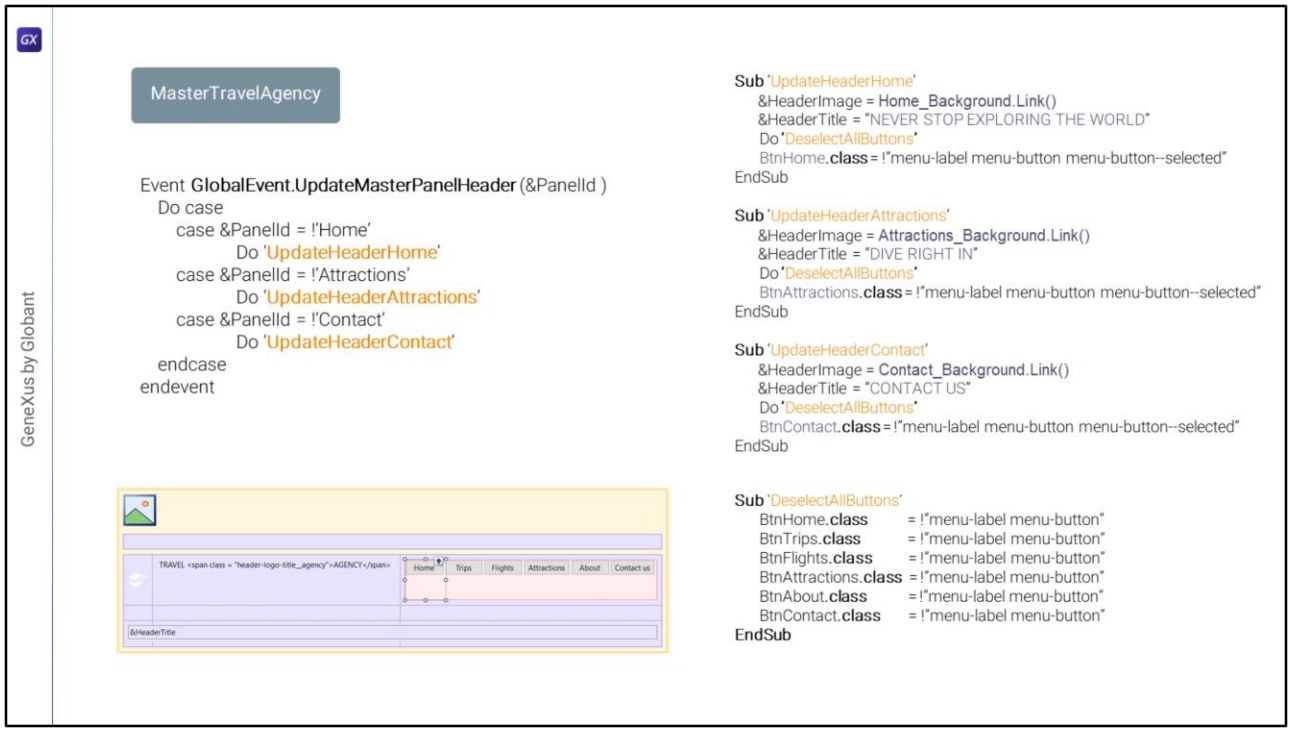


As classes podem ser atribuídas aos controles tanto de forma estática, através da propriedade do controle, como dinâmica, em qualquer lugar onde seja aceito código, como nos eventos.



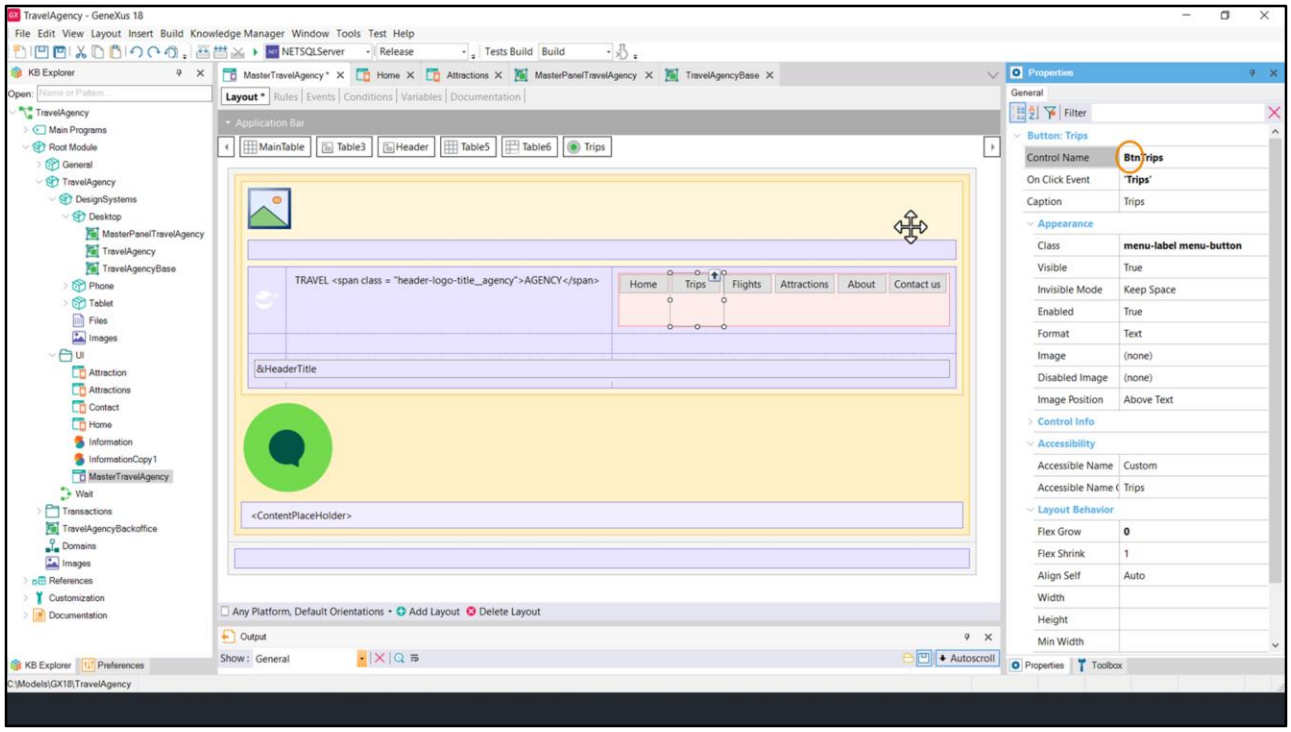
Dessa maneira, poderíamos escrever uma sub-rotina “DeselectAllButtons”, que vemos que o que faz é atribuir as duas classes, menu-label e menu-button, a todos os botões do menu.

Então, o que faríamos nessa sub-rotina da qual partimos seria primeiro invocar essa outra para deixar todos os botões como desselecionados. E depois, simplesmente atribuir ao botão correspondente, que neste caso é o de nome BtnHome, suas classes, que são as mesmas dos outros, mais a menu-button--selected.



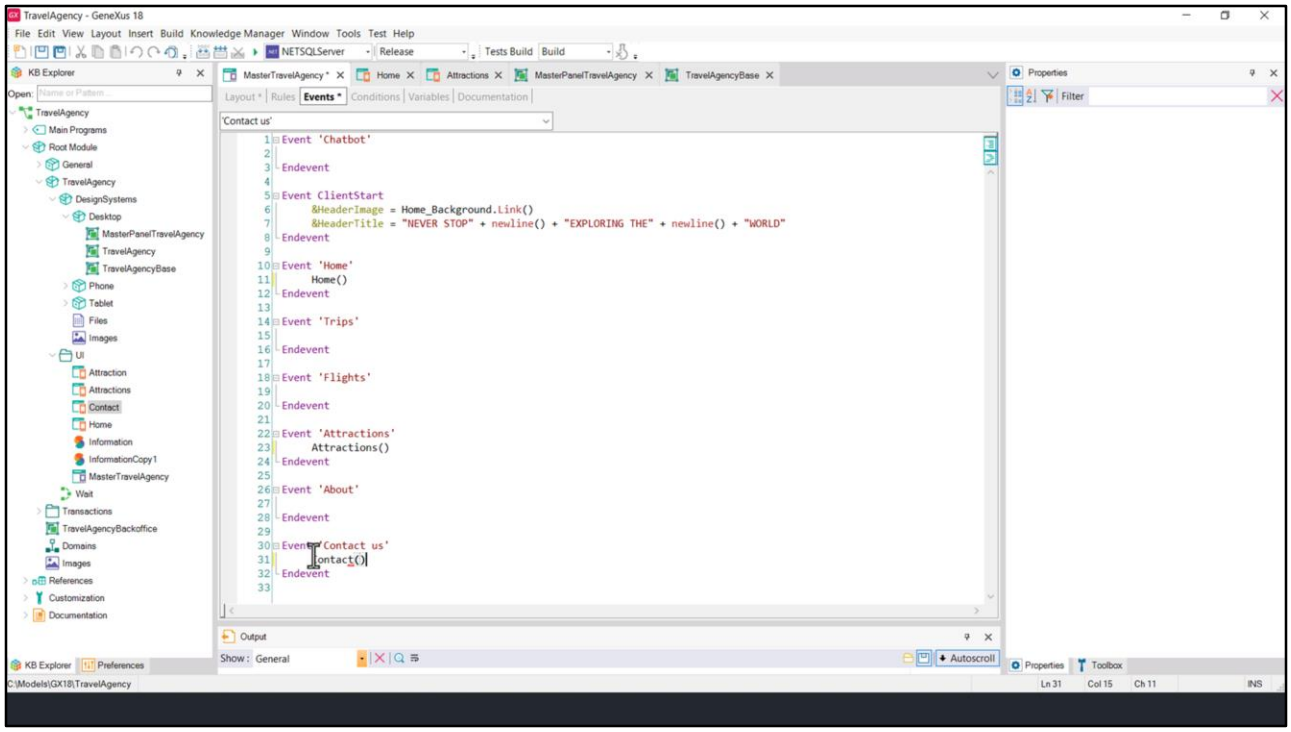
Assim ficará então a aba de eventos do Master Panel para implementar corretamente o Header quando o Master Panel estiver sendo chamado pelos painéis Home, Attractions ou Contact. Vejam que as sub-rotinas para os dois últimos têm exatamente a mesma lógica que vimos para Home: carregam a imagem correspondente, o título e desmarcam todos os botões, para depois deixar selecionado apenas o que corresponde.

Falta agora implementar a carga correta do Header quando quem chama é Attraction, mas antes vamos levar tudo isso para o GeneXus, para ir em segurança.

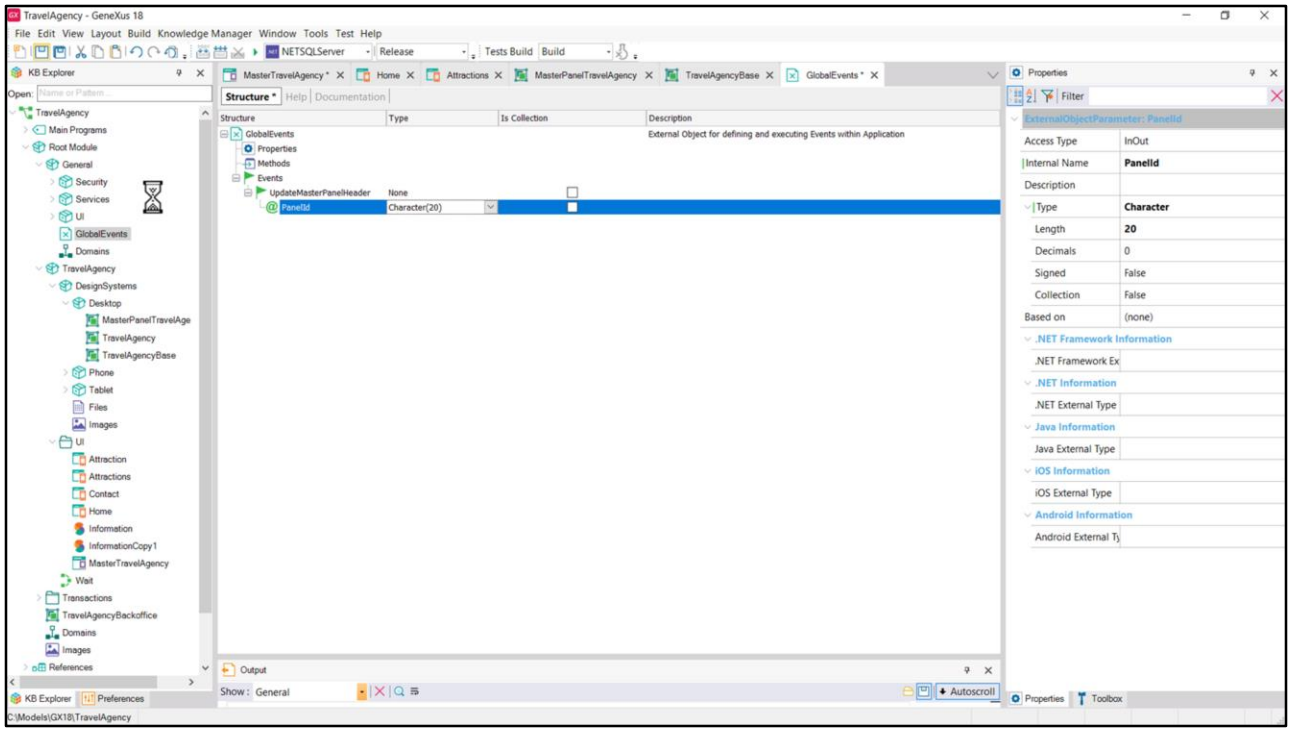


Aqui temos o Master Panel. Lembremos que, devido a um bug no editor, os botões ficaram invisíveis. Por enquanto, para torná-los visíveis novamente e poder operar com eles, modifiquemos esta propriedade, que era a que estava relacionada com o bug. Depois, deixaremos como deve ser, com o valor Center.

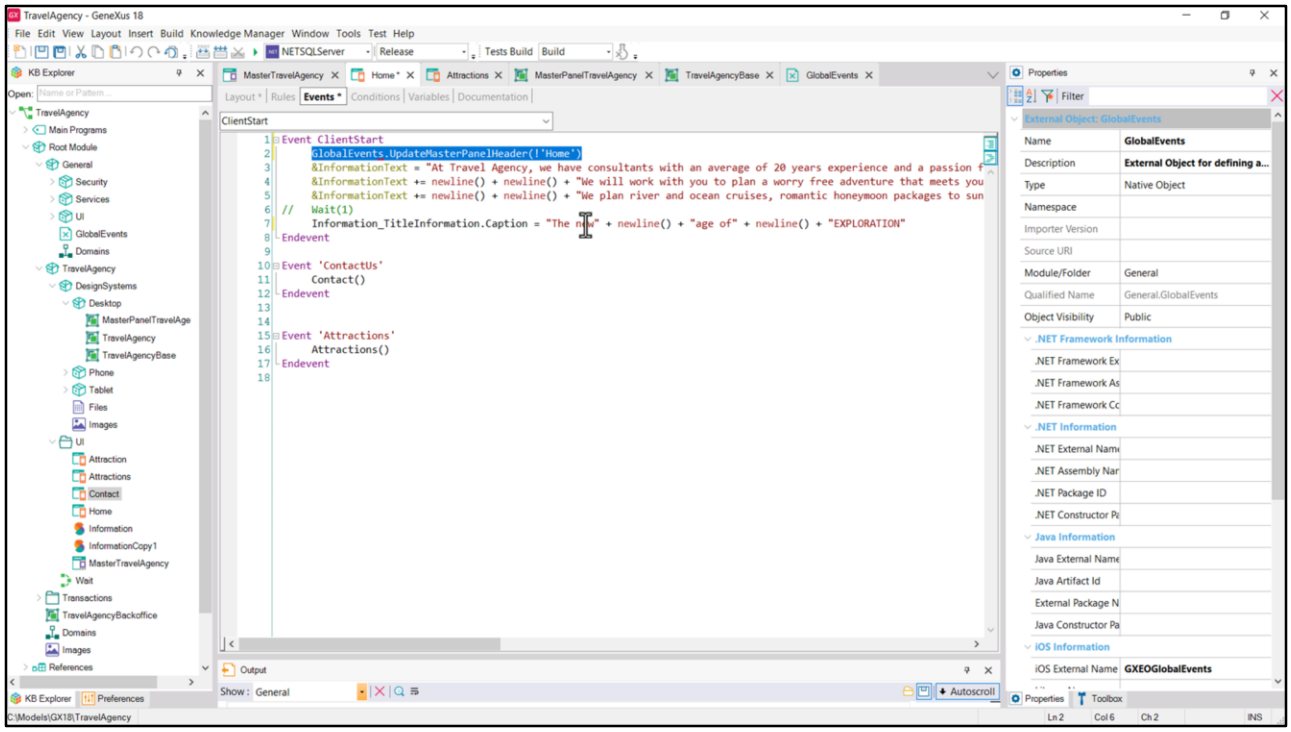
Aproveitemos e alteremos o nome dos botões para precedê-los com “Btn”, assim ficará mais claro seu uso posteriormente no código.



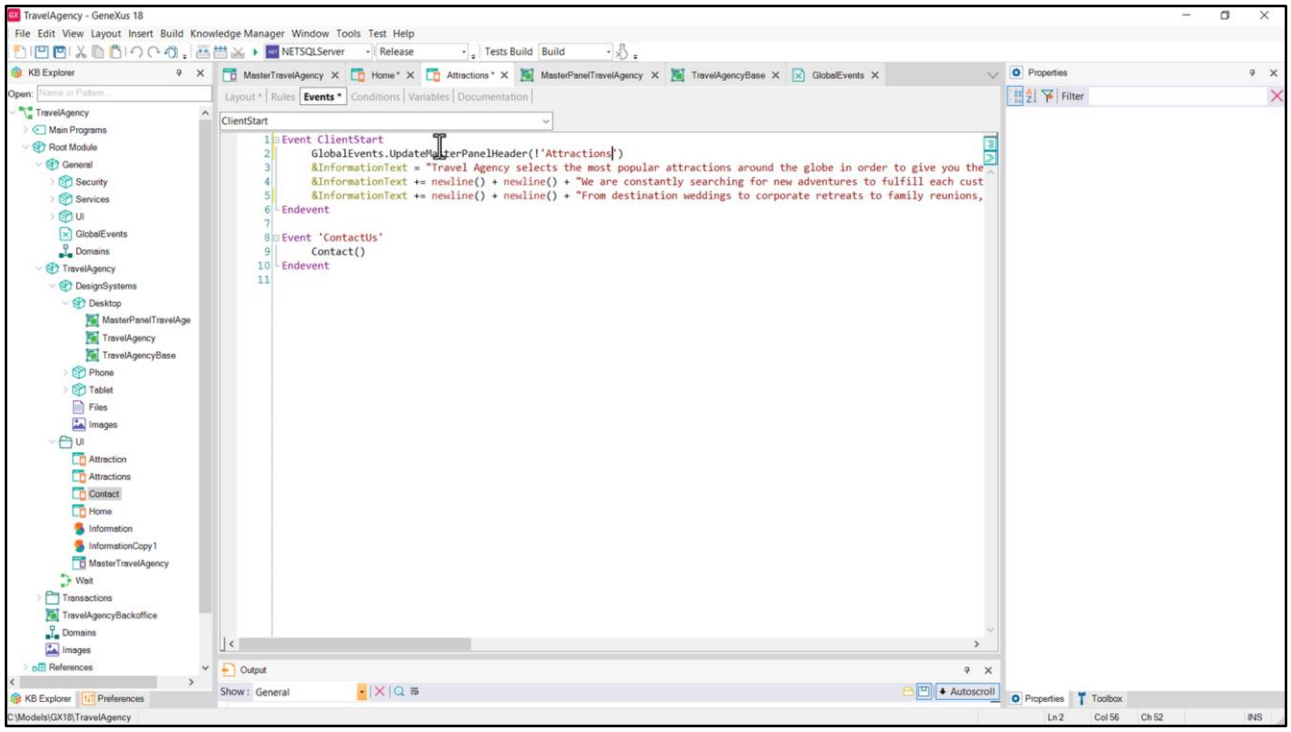
Comecemos por realizar as invocações em cada botão aos painéis. Assim, quando for pressionado o botão de Home, deveremos invocar o panel Home. Quando for pressionado Attractions, o de Attractions, e no de Contact us, o panel Contact. Os outros painéis, por enquanto, ainda não estão nem sequer desenhados.



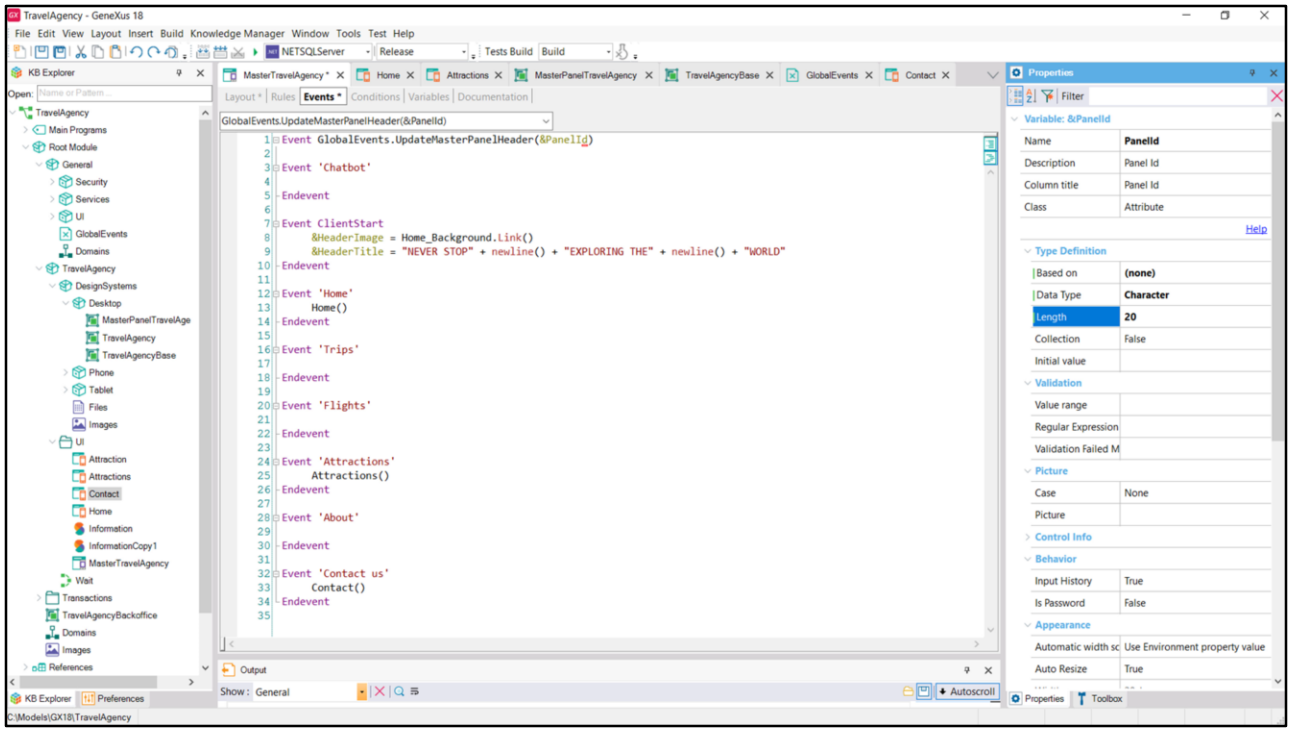
Agora editemos o objeto GlobalEvents. Adicionemos o evento UpdateMasterPanelHeader, com variável &PanelId de tipo Character.



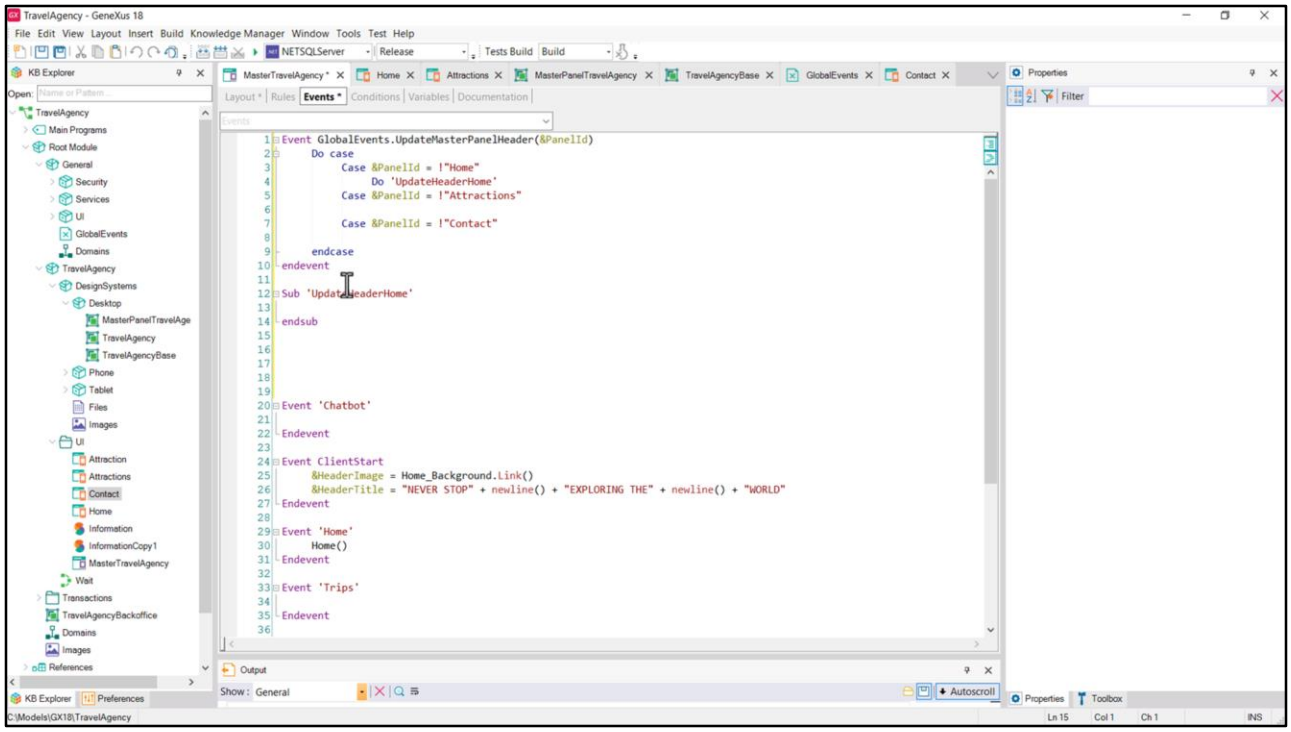
Vamos para Home e no ClientStart o disparamos. Vou colocá-lo melhor no início do código para que fique mais claro.



E o mesmo fazemos em Attractions, mudando aqui para ‘Attractions’. E em Contact.

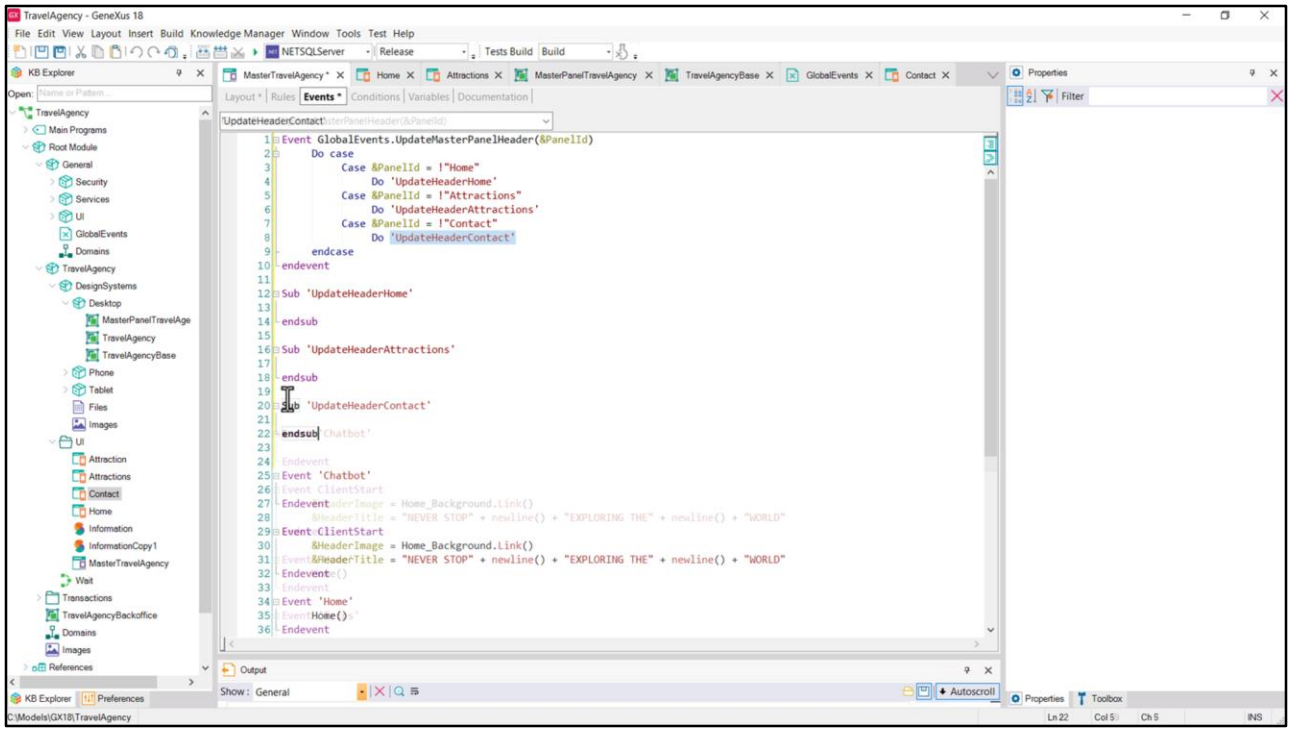


Agora programaremos o evento global no Master Panel… Definamos a variável… &PanelId… não queremos que seja baseada no domínio Id, mas sim que seja do tipo Character.

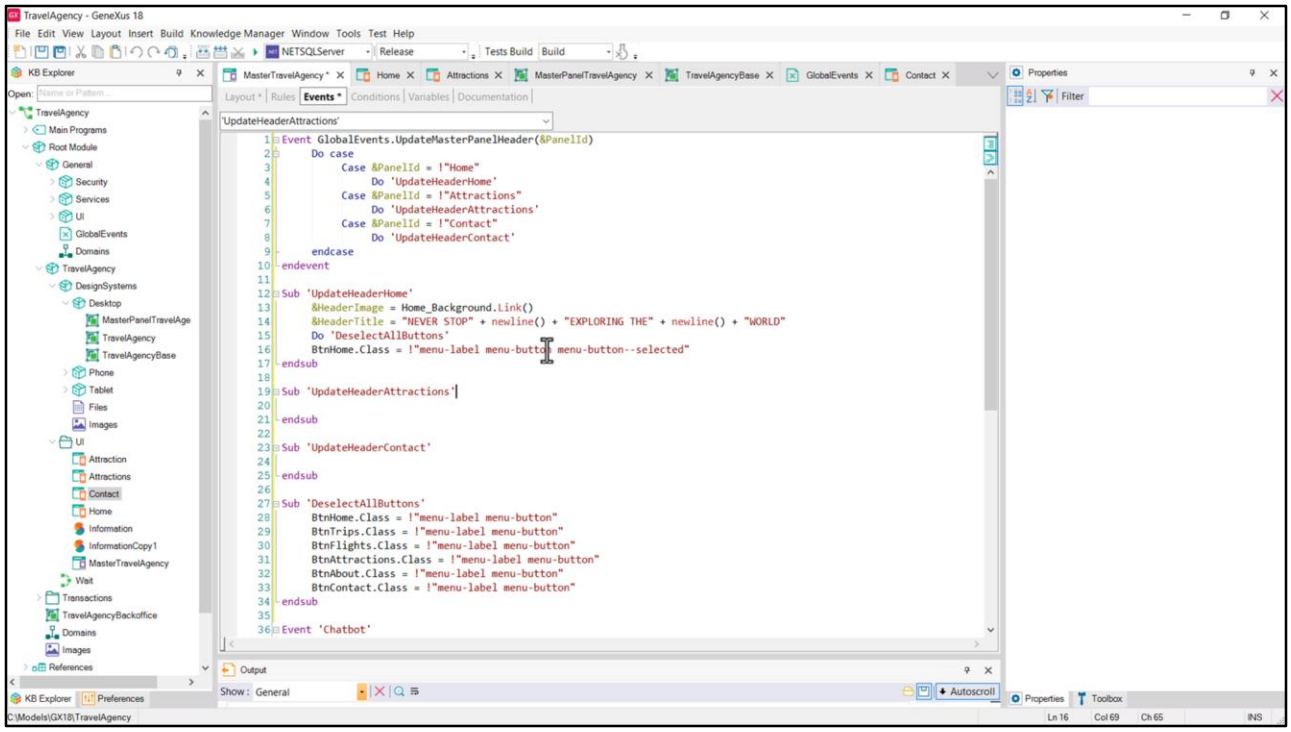


Bem, adicionamos o comando **Do case**, para poder tomar as diferentes ações com base no valor da variável... Se for ‘Home‘ será feita uma coisa, se for ‘Attractions‘ outra, se for ‘Contact‘ outra.

O que será feito se for ‘Home‘? Invocar esta sub-rotina, que precisamos definir... Podemos especificá-la em qualquer lugar desta aba de eventos, que como sabemos é declarativa, não importa a ordem em que definimos os eventos e as sub-rotinas.

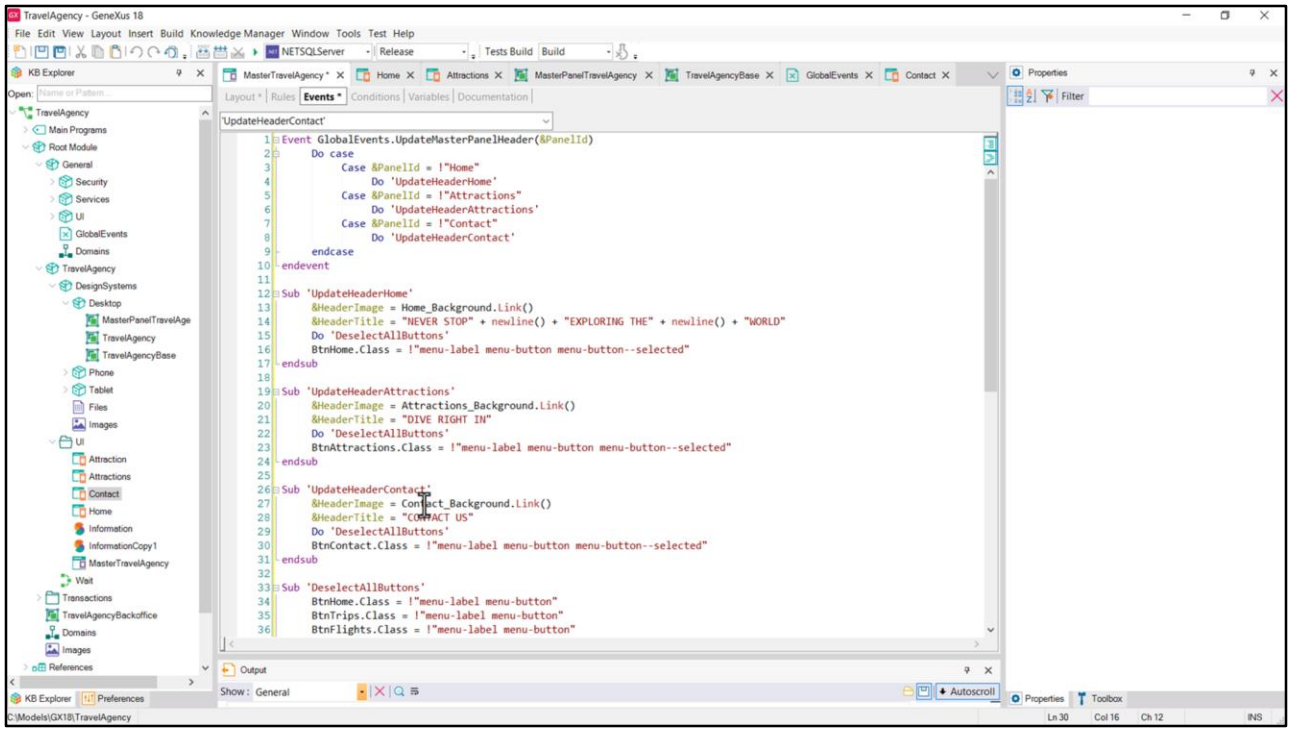


Se o valor que assume a variável for ‘Attractions‘, chamaremos esta outra sub-rotina. E se for ‘Contact‘, esta outra.



Programemos a primeira. À variável &HomeImage atribuímos a imagem que havíamos inserido na KB na etapa de preparação, método link.
Ao título este aqui.
Depois invocamos a sub-rotina DeselectAllButtons que definiremos em seguida. E, por último, ao botão Home atribuímos as 3 classes.

Agora especifiquemos a sub-rotina DeselectAllButtons, na qual a todos os botões do menu atribuímos as duas primeiras classes.

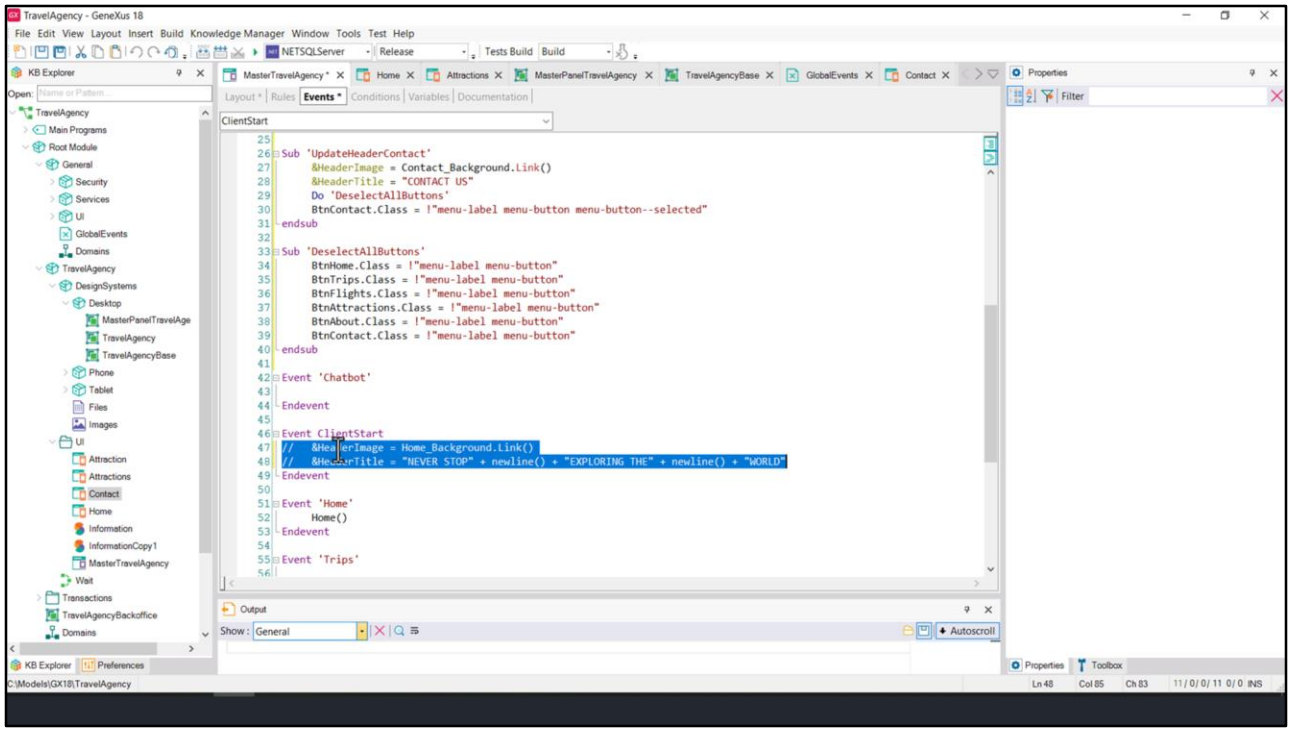



```
Event GlobalEvents.UpdateMasterPanelHeader(&PanelId)
    Do case
        Case &PanelId = !"Home"
            Do 'UpdateHeaderHome'
        Case &PanelId = !"Attractions"
            Do 'UpdateHeaderAttractions'
        Case &PanelId = !"Contact"
            Do 'UpdateHeaderContact'
    endcase
endevent

Sub 'UpdateHeaderHome'
    &HeaderImage = Home_Background.Link()
    &HeaderTitle = "NEVER STOP" + newline() + "EXPLORING THE" + newline() + "WORLD"
    Do 'DeselectAllButtons'
    BtnHome.Class = !"menu-label menu-button menu-button--selected"
endsub

Sub 'UpdateHeaderAttractions'
    &HeaderImage = Attractions_Background.Link()
    &HeaderTitle = "DIVE RIGHT IN"
    Do 'DeselectAllButtons'
    BtnAttractions.Class = !"menu-label menu-button menu-button--selected"
endsub

Sub 'UpdateHeaderContact'
    &HeaderImage = Contact_Background.Link()
    &HeaderTitle = "CONTACT US"
    Do 'DeselectAllButtons'
    BtnContact.Class = !"menu-label menu-button menu-button--selected"
endsub

Sub 'DeselectAllButtons'
    BtnHome.Class = !"menu-label menu-button"
    BtnTrips.Class = !"menu-label menu-button"
    BtnFlights.Class = !"menu-label menu-button"
```

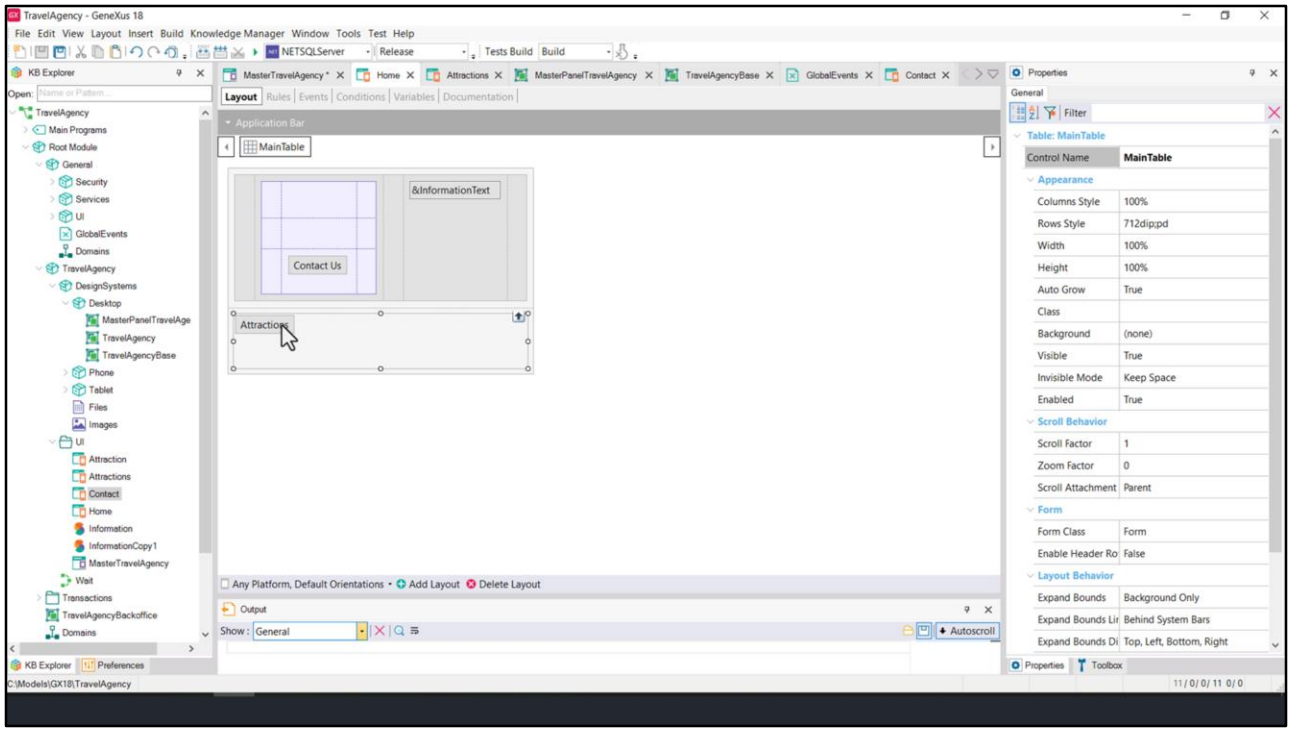
Analogamente, programamos as outras duas sub-rotinas…
Aqui a imagem se chamava assim… e o texto era este… (o extraímos do Figma)… e o botão aqui é o de Attractions.

E para Contact também fazemos as substituições necessárias.

Ao gravar nos dá um erro… não está entendendo o nome BtnContact. Se verificarmos… é que se chamava assim. Vamos remover este final. Agora sim.



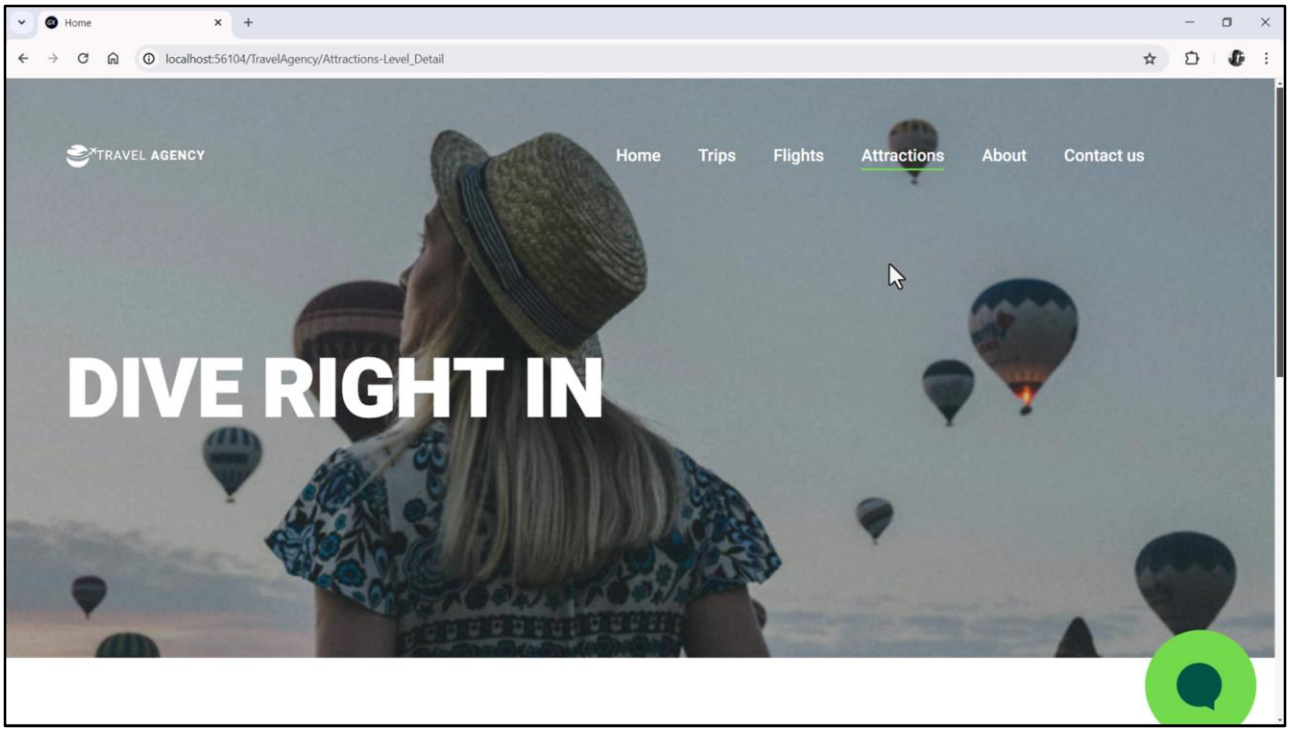
Esta carga do ClientStart do Master Panel já não será necessária.



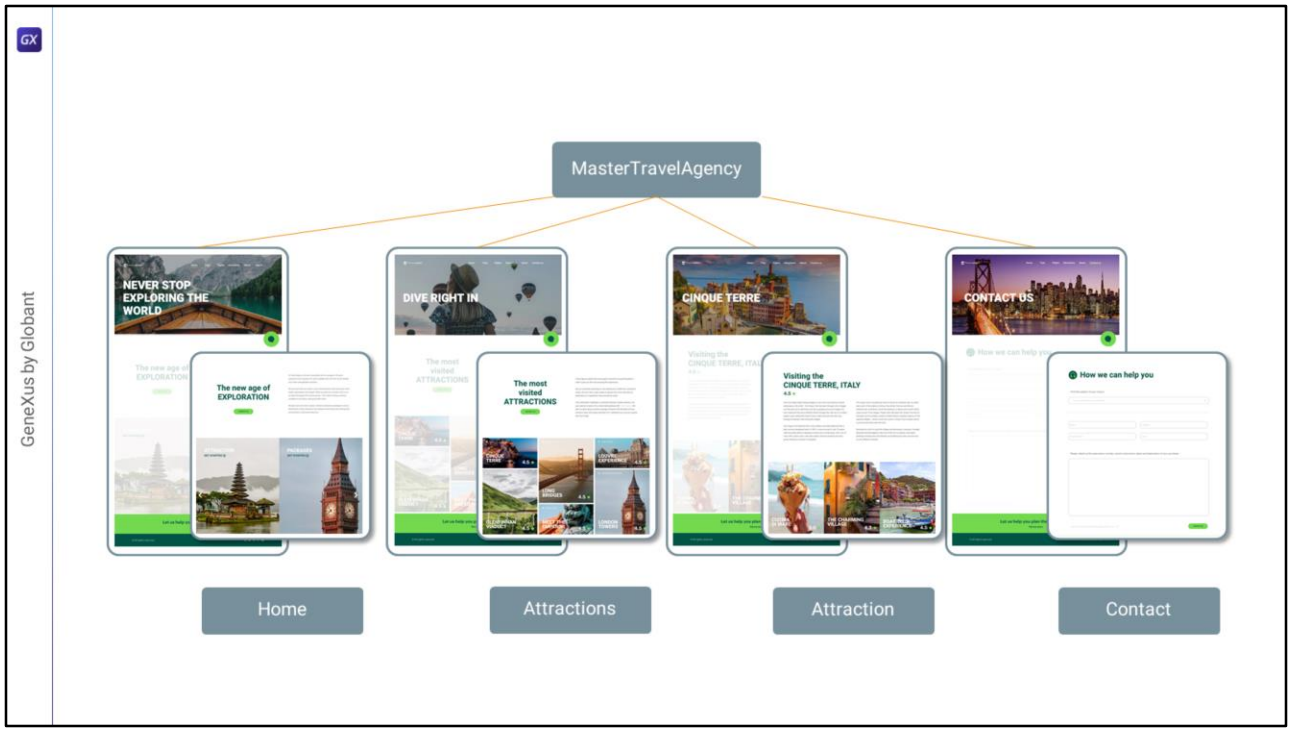
Assim como este botão também não para chamar o panel de atrações.

Aqui observemos tudo... está tudo bem.

Voltamos a colocar a propriedade Align Items do flex em Center. E Executemos.



Vemos perfeita a página Home... e se clicarmos aqui, perfeita a de atrações, e aqui a de contato. Voltamos para Home... está tudo perfeito.



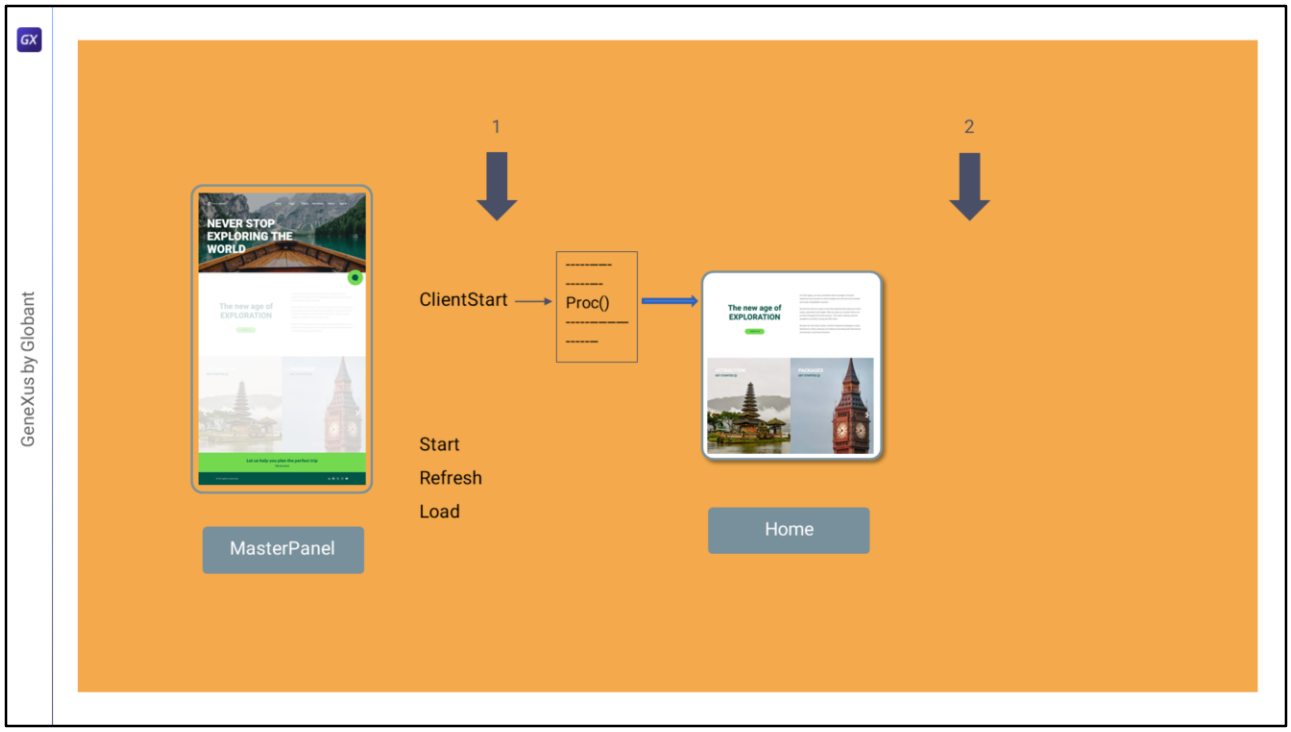
Por enquanto (e isso está em processo de mudança, então dependerá de quando você assistir a este vídeo se isso continua sendo assim ou não), toda vez que um painel é executado, seu Master Panel é executado novamente, não importando se entre execuções de painéis o Master Panel é o mesmo. Ou seja, ele é construído e destruído entre navegações, a cada vez.

Isso certamente não é o mais eficiente do mundo e pode trazer complicações como piscadas, por exemplo.

Para entender um pouco melhor tudo isso, vou contar como funciona no momento.

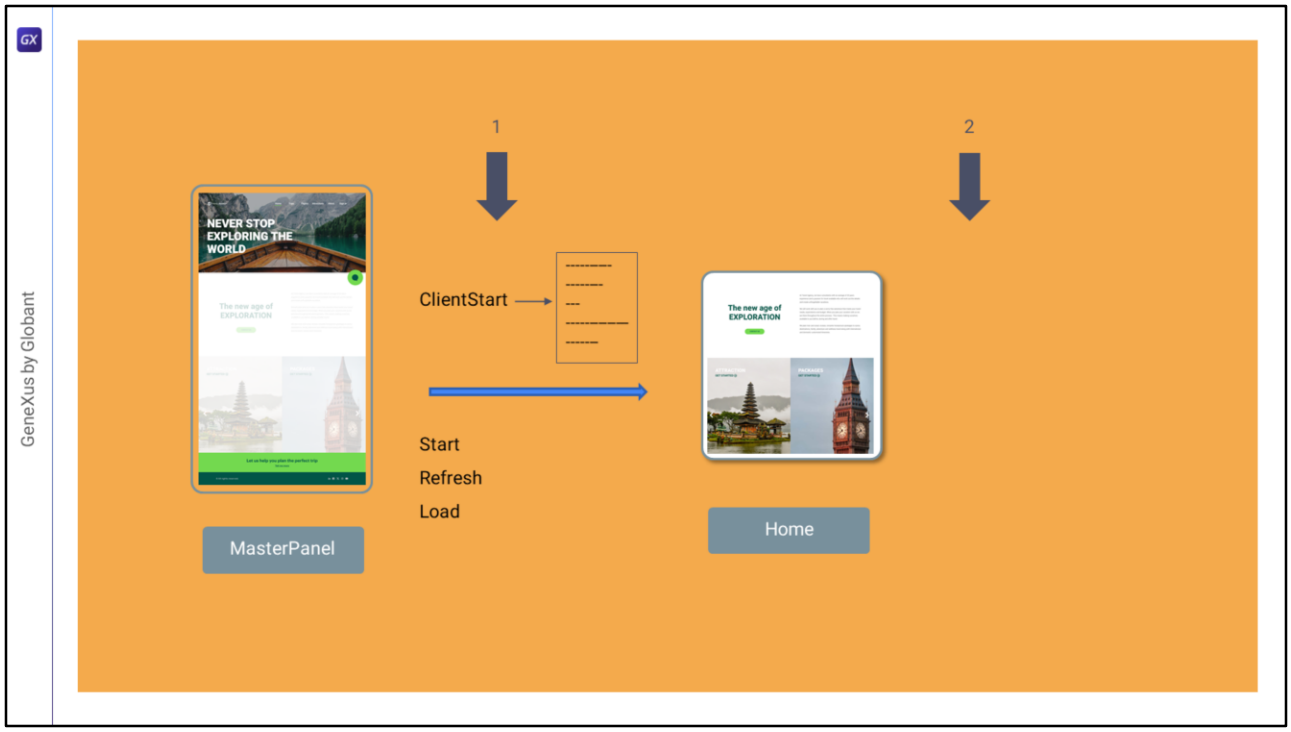
Ao lançar a execução do panel Home, por exemplo, a ordem de execução será a seguinte:
Primeiro, constrói-se o Master Panel, onde são atribuídas todas as propriedades estáticas dos controles do layout (as que foram indicadas na janela de propriedades de cada controle; são as que costumamos chamar de propriedades em tempo de design), e em seguida é executado o evento ClientStart.
Se o evento ClientStart tiver alguma invocação a um procedimento, por exemplo, que necessariamente deverá ser resolvido no servidor, ou a um Data Provider, sua execução é disparada de forma assíncrona, e é renderizado pela primeira vez o Master Panel com o que se tem até o momento até aqui. Ou seja, não se espera que esta Proc termine para continuar e só ao final do evento renderizar. Renderiza-se e depois continua.
Por outro lado, se dentro do ClientStart não houver nenhuma invocação assíncrona, então, nesse caso, a primeira renderização ocorre apenas quando o ClientStart termina.
Seja qual for o caso, após a primeira renderização do Master Panel, já se inicia a construção do Panel projetado no ContentPlaceHolder, ou seja, ANTES de serem executados os eventos Start, Refresh e Load.

Seja porque estamos aqui, seja porque estamos aqui



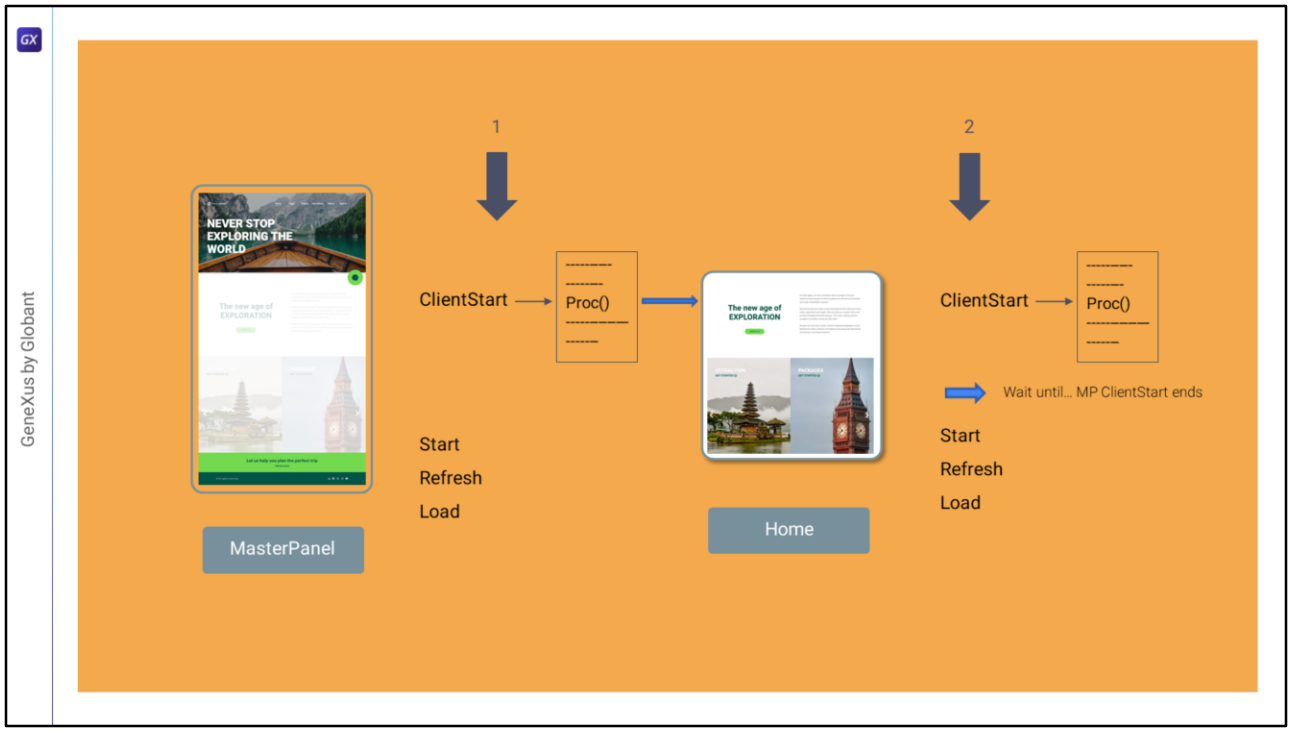
Abre-se, então, outra linha de execução concorrente.

Na que estávamos, aqui já havia sido renderizado pela primeira vez o Master Panel e iniciada a construção do Panel. Quando a Proc terminar, continua a execução do restante do ClientStart, e se houve na Proc ou aqui alguma alteração na User Interface, volta a ser renderizado. Em seguida, são executados os demais eventos do sistema (claro, renderizando caso introduzam alterações na UI).



E se não havia Proc nem nada assíncrono, então aqui já havia sido renderizado o Master Panel e iniciada a construção do Panel, de modo que agora são executados os demais eventos do sistema.

E o que acontece na linha concorrente?

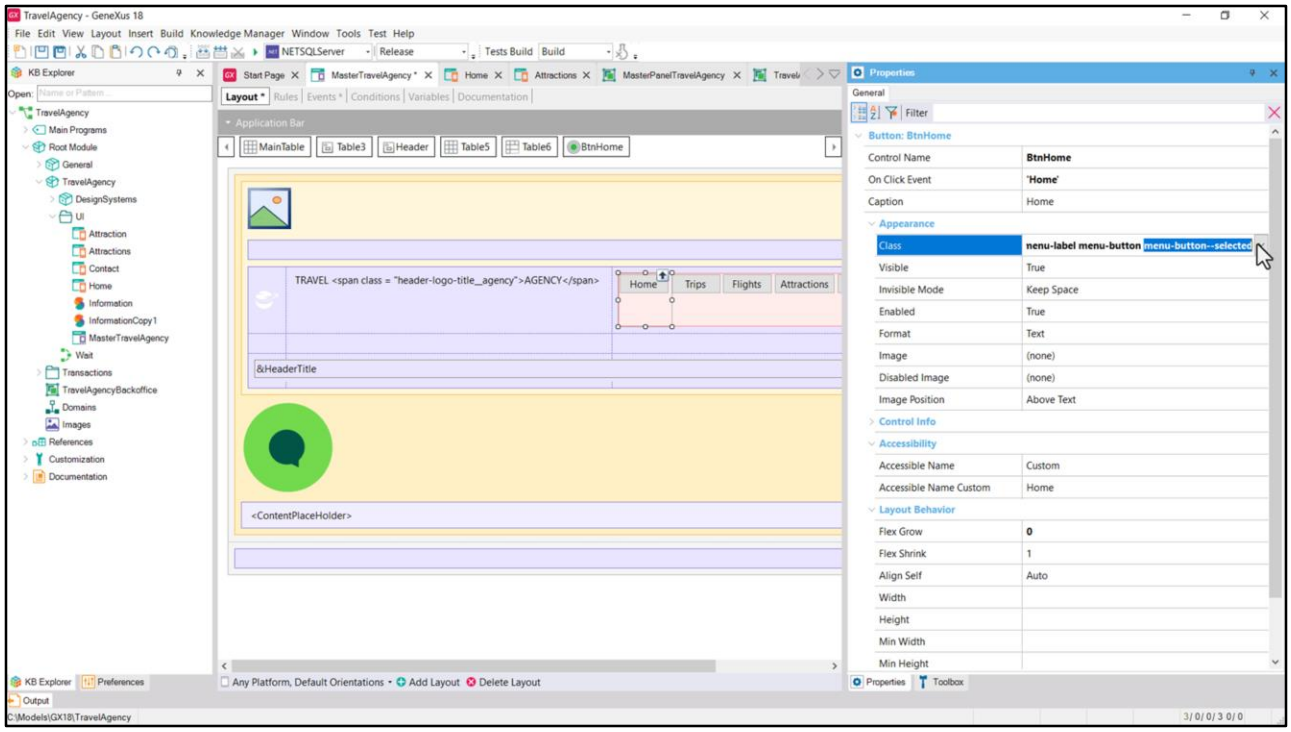


Vejamos no primeiro caso:

- É construída a User Interface do panel com base nas propriedades definidas em Tempo de Design (ou seja, as estáticas),
- e começa a executar o evento ClientStart, que, novamente, se tiver alguma execução assíncrona, nesse momento é renderizado o panel, e espera-se que termine a execução assíncrona. Quando esta termina, continua com a execução do ClientStart e, ao final, se for necessário, é renderizado novamente.
- Se não havia execução assíncrona, então é renderizado pela primeira vez o panel ao finalizar o evento ClientStart.
- E aqui prestem atenção: se o ClientStart do Master Panel não terminou quando este sim o fez, então aqui espera-se até que o faça, para só então começar a executar o Start, e depois o Refresh e, em seguida, o Load.

Em resumo, o Start do Panel não começa até que o ClientStart do Master Panel tenha terminado.

Sigamos essa ordem em detalhe para o nosso caso, prestando atenção ao que acontece com a imagem e o título do Header e os botões do menu.

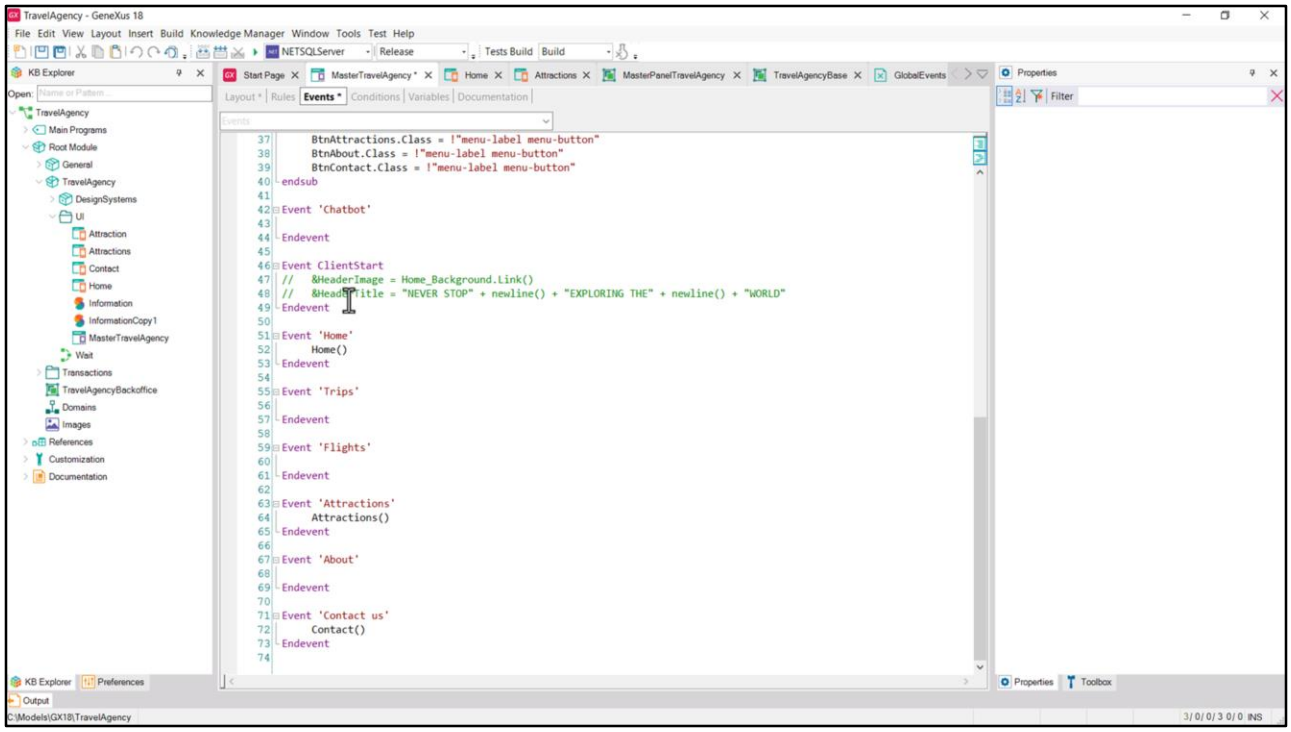


Então, começa-se pelo Master Panel, como dizíamos, e a primeira coisa que se faz é aplicar as propriedades estáticas.

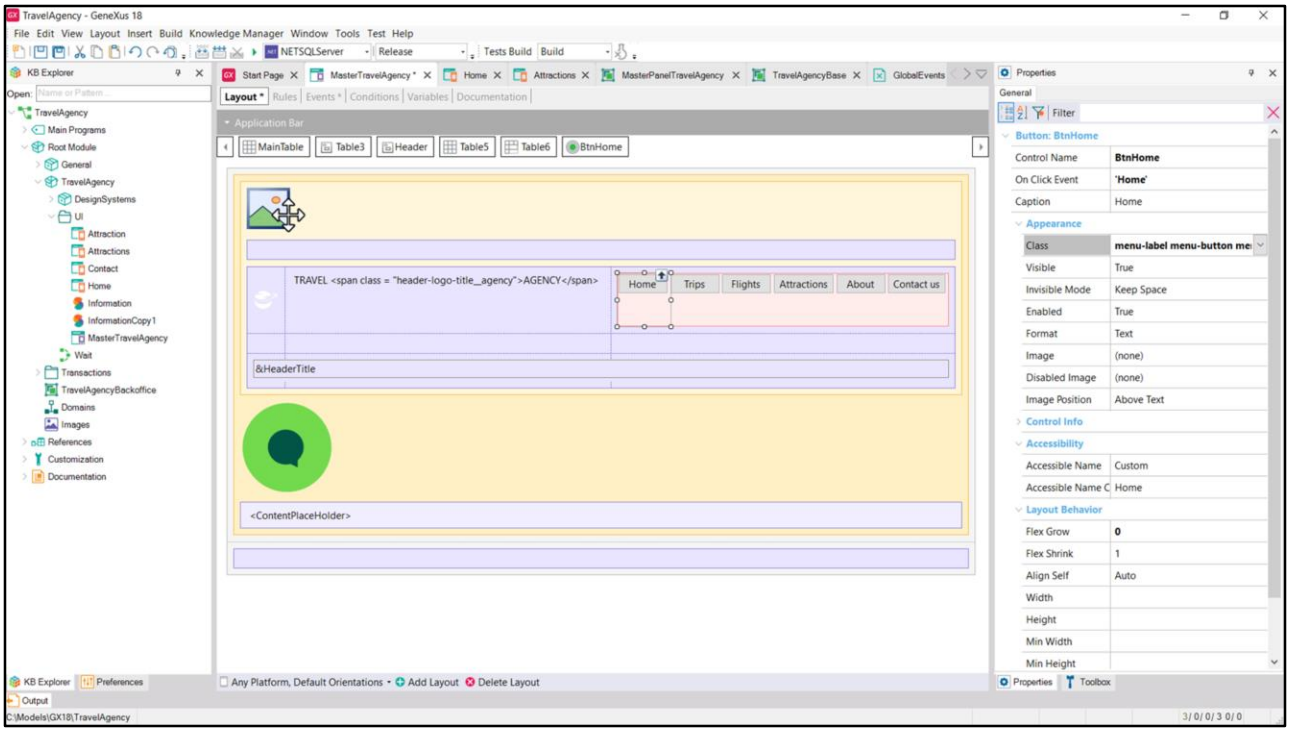
Então, as variáveis de imagem e título estarão vazias em conteúdo, terão suas propriedades, as propriedades, por exemplo, a das classes.

E o menu também já terá associadas a cada botão todas essas classes. Em particular, observemos que para o botão do Home temos também a classe selecionada, que havíamos definido no vídeo anterior (havíamos deixado estática porque ainda não tínhamos começado a implementar toda essa questão dinâmica de poder trocar de tela, certo?, de acordo com o menu). Teremos que removê-la, claramente. Mas, por enquanto, quero deixá-la para que pensemos por que teria que removê-la.

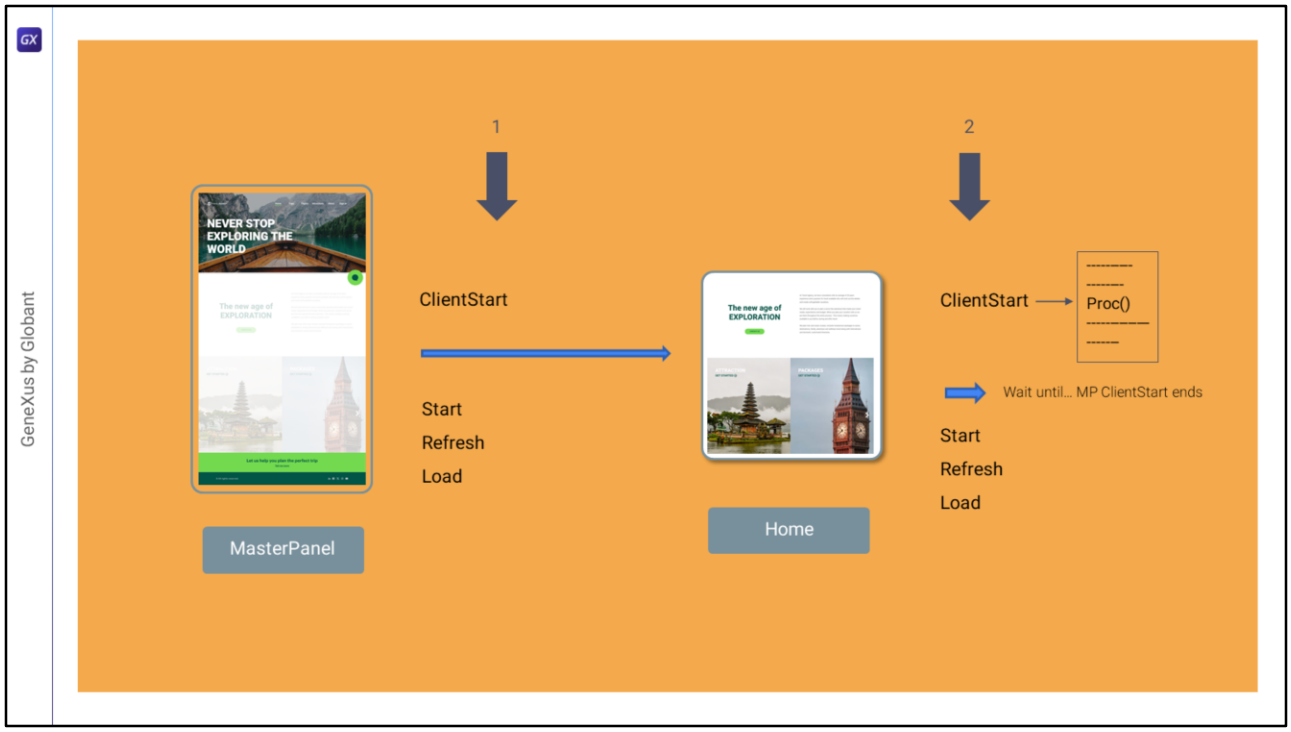
Dizíamos, então, neste momento são consideradas todas as propriedades do tempo de design. Em particular, portanto, são atribuídas as classes definidas estaticamente aos nossos controles.



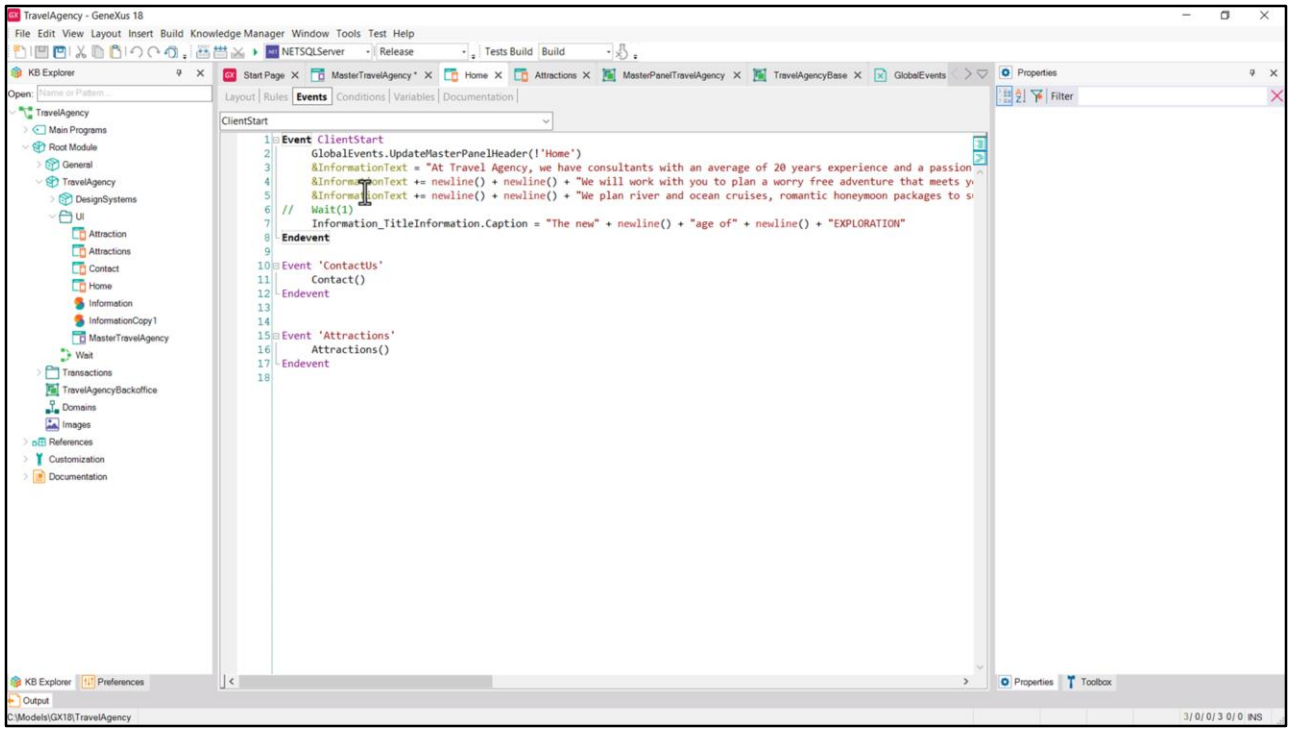
Bem, depois será executado, dizíamos, o evento ClientStart, que tínhamos comentado... então ali não acontece nada.



E, como consequência de tudo isso, é renderizado o Master Panel com o que se tem até o momento: as variáveis de imagem e título estarão vazias, e o menu com a opção Home selecionada.



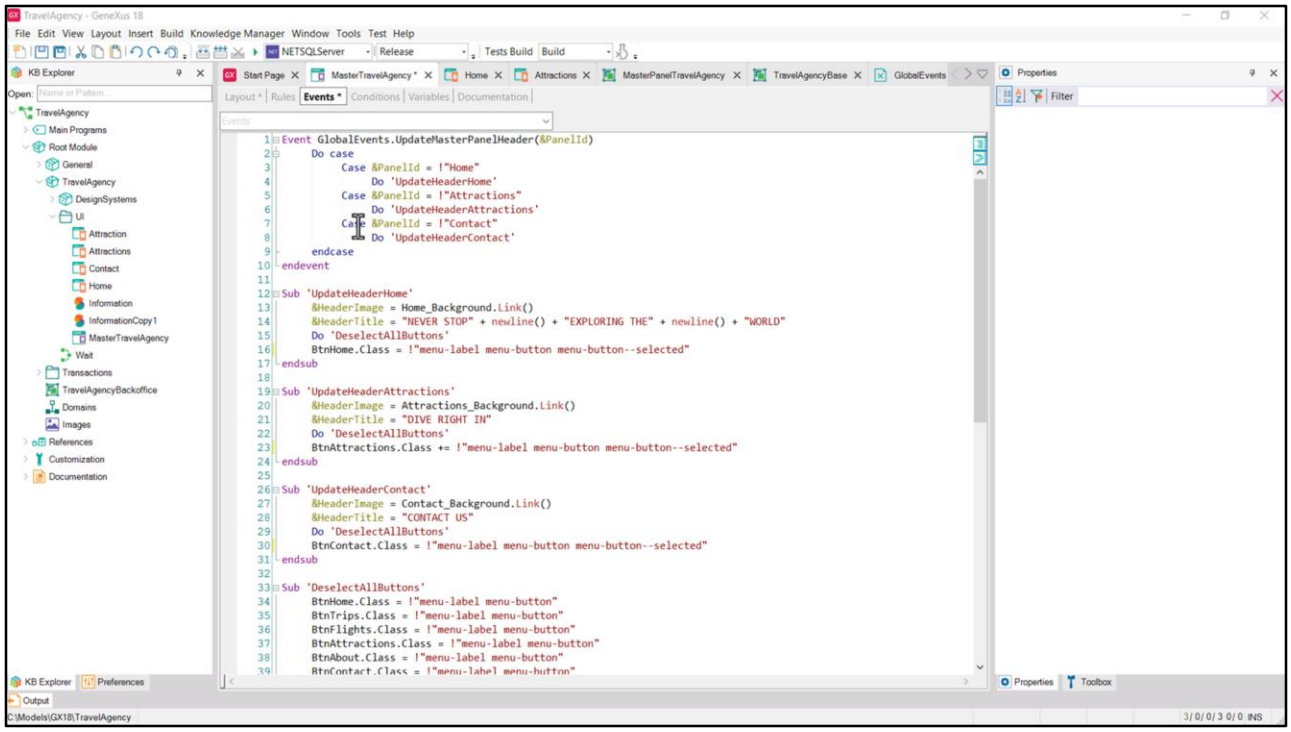
Depois é executado o Panel propriamente dito, e seu evento ClientStart.



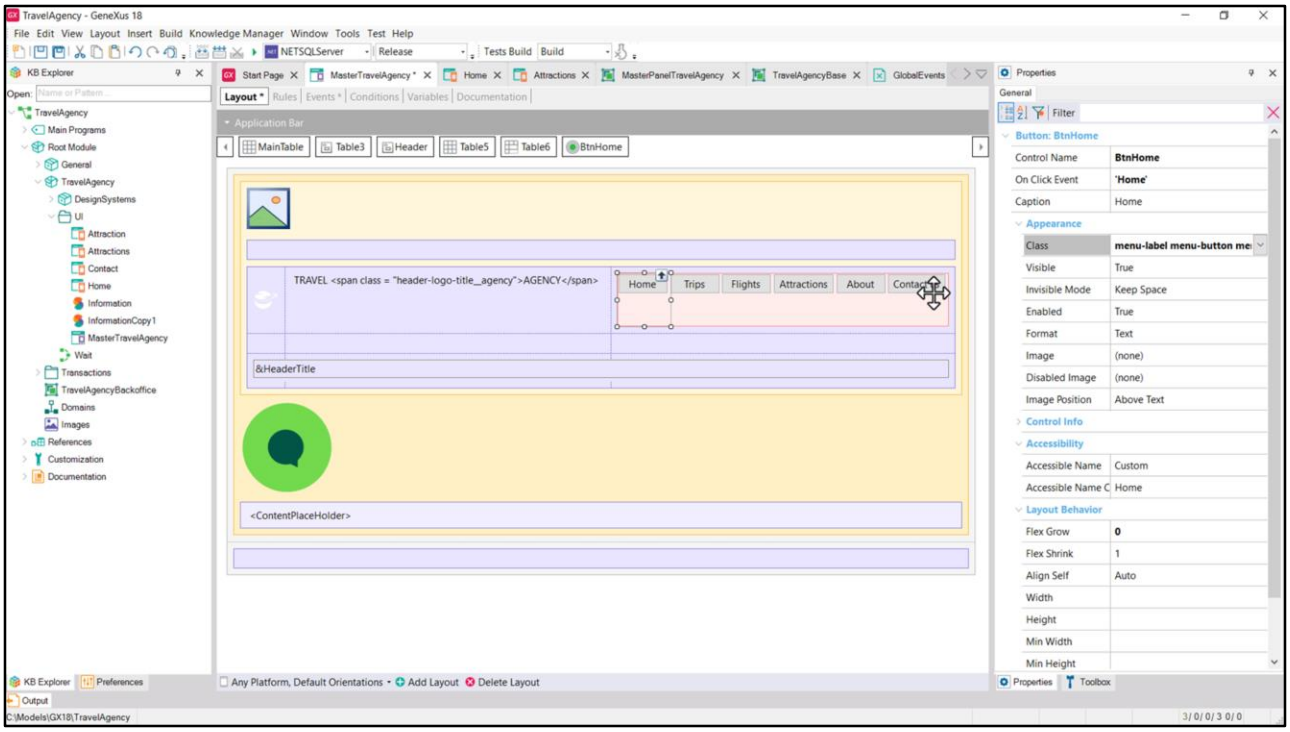
Então vai ser executado o evento ClientStart…

E se o observarmos, ele não só tem a invocação ao evento global, mas também estas atribuições.

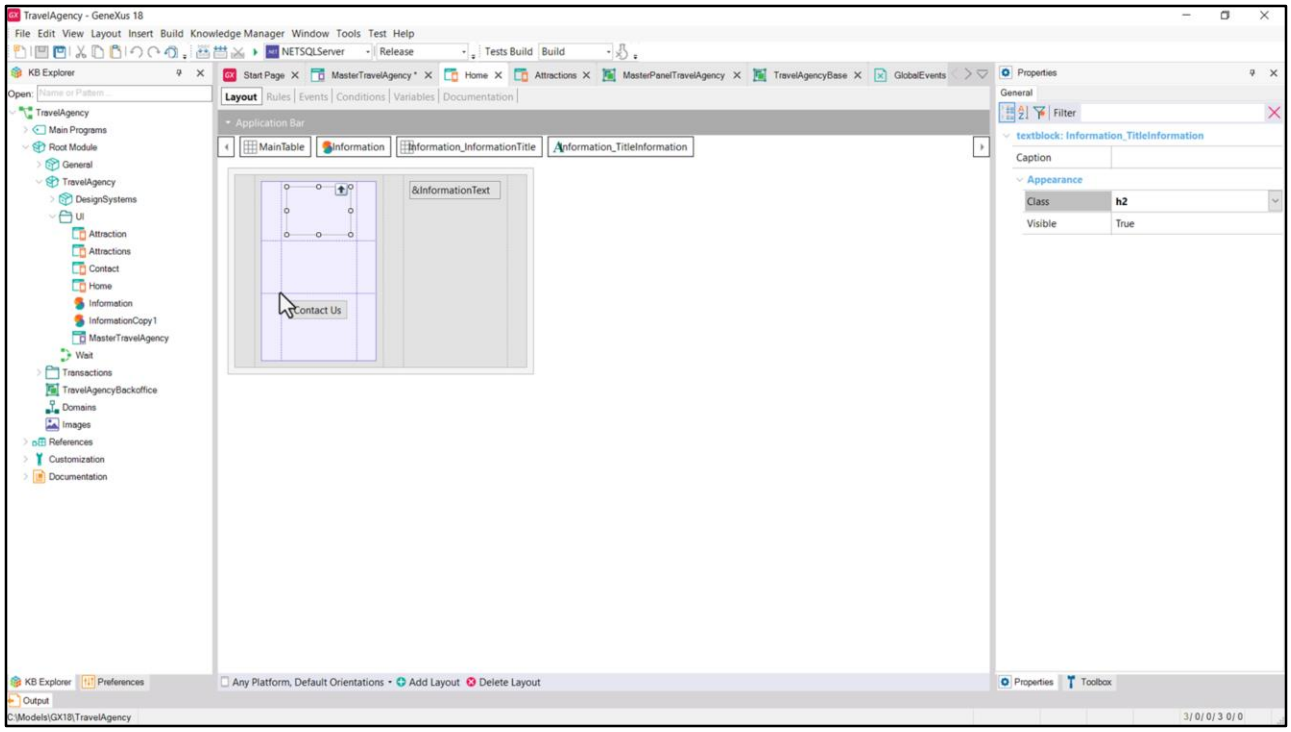
Tive minhas dúvidas, na ocasião, sobre colocar a invocação ao evento global no início ou no final do código. O resultado será o mesmo, dado que o que vai acontecer é que será disparado o evento global de maneira assíncrona, mas sem bloquear a execução, ou seja, não vai parar e esperar que termine para prosseguir com este próximo comando, mas continuará até finalizar.



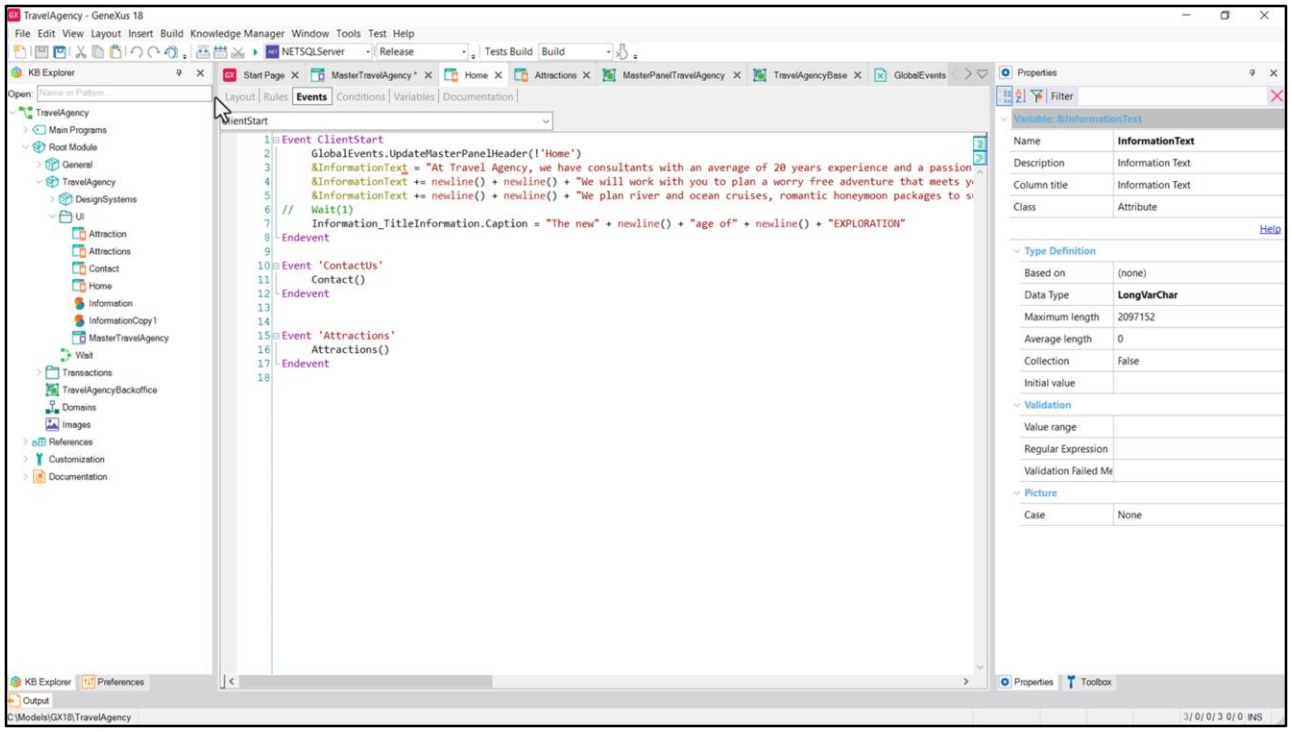
Como o evento global está sendo escutado pelo Master Panel, que está ativo no cliente nesta execução, então executará seu código.



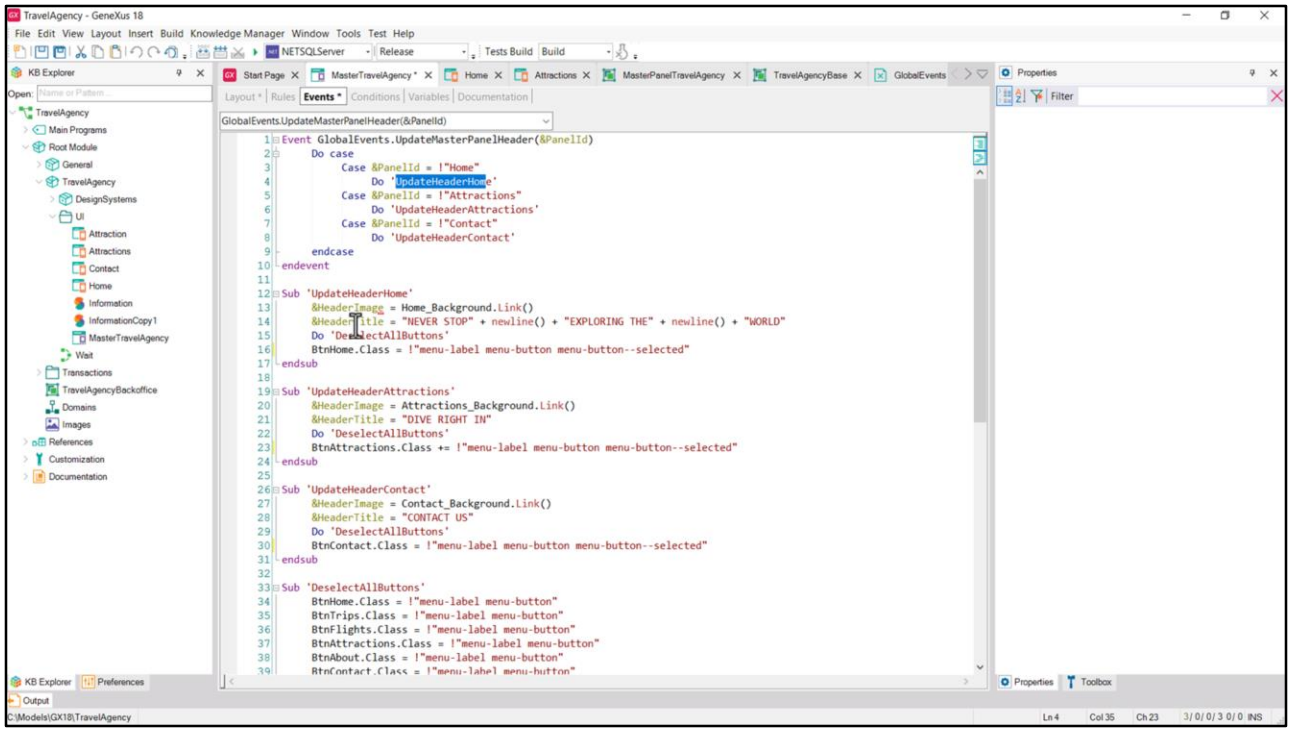
Assim, resumindo o estado da execução até este momento: foi renderizada a tela do Master Panel primeiro, com a imagem vazia e o texto vazio, mas as classes para os botões do menu aplicadas, de modo que o menu estaria com Home destacado.



Imediatamente foram renderizadas estas partes do layout de Home…

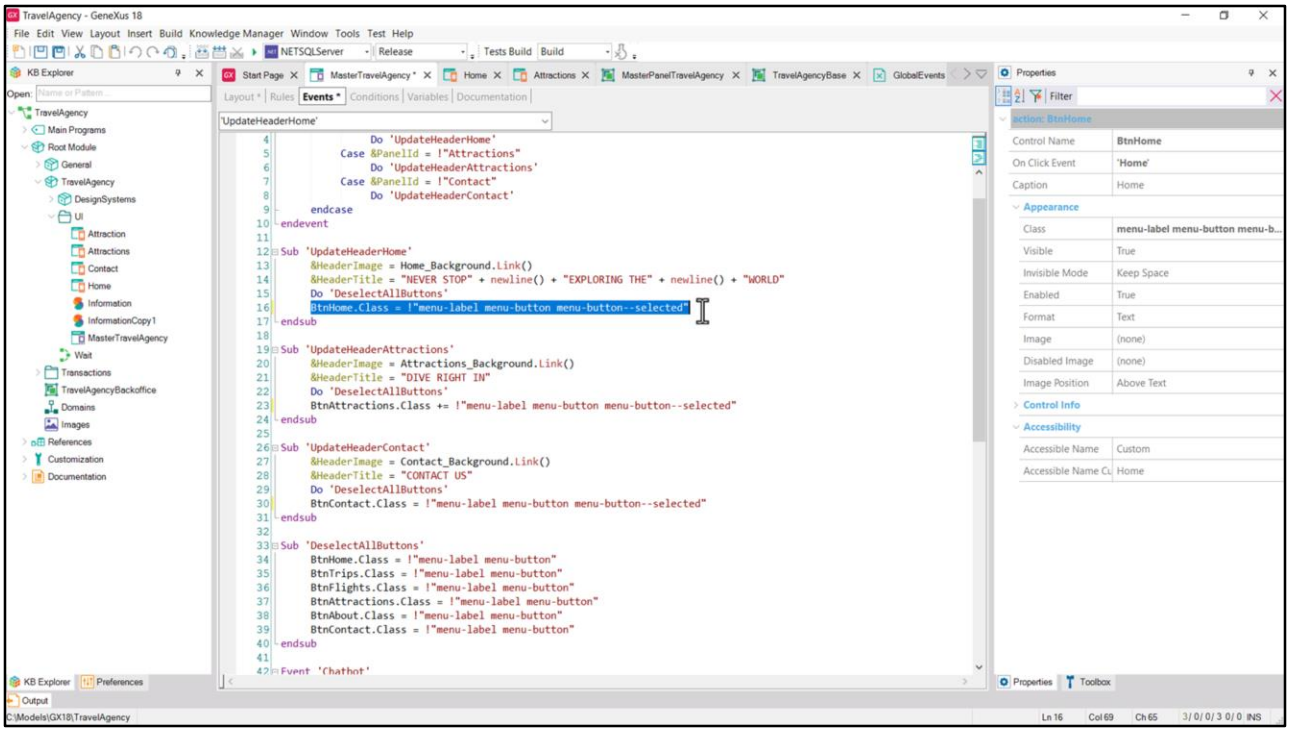


…e neste exato momento…



…está sendo executado o Do case do evento global, evento que sempre é do cliente.

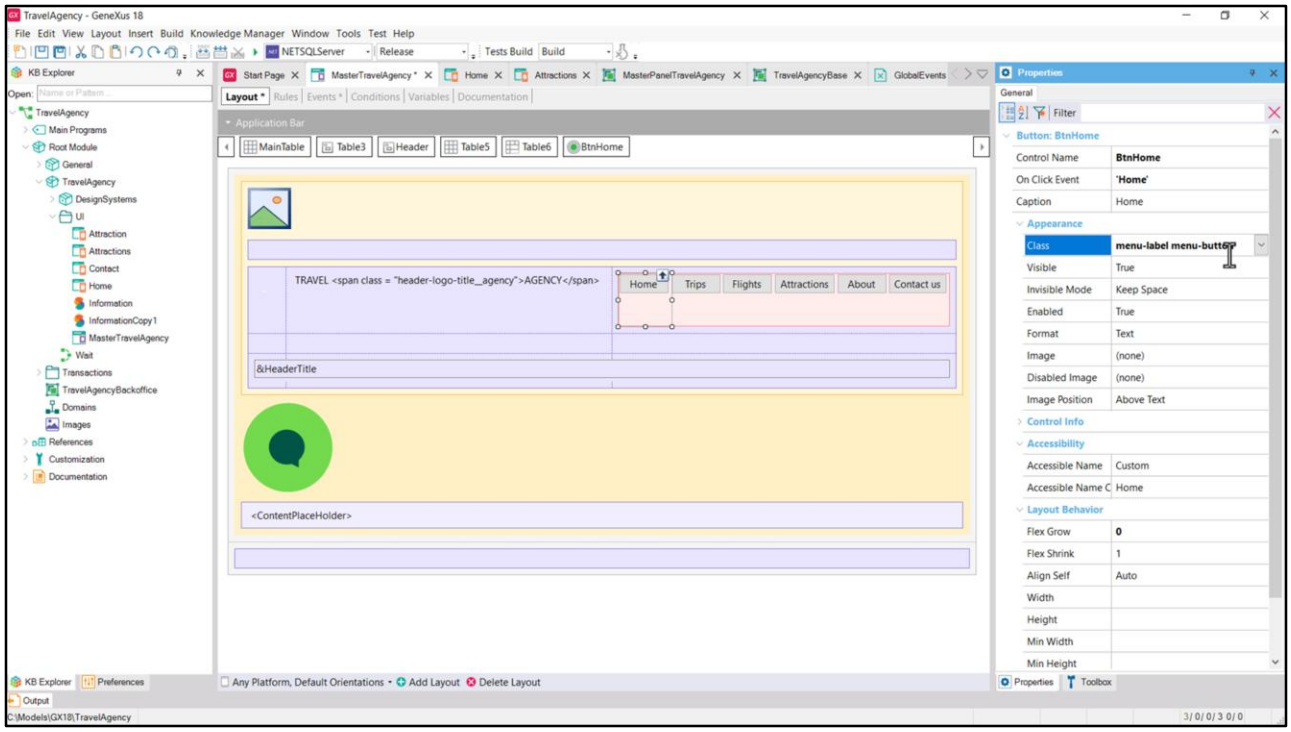
Então aqui é executada esta sub-rotina… que atribui valor, agora sim, às variáveis &HeaderImage e &HeaderTitle.



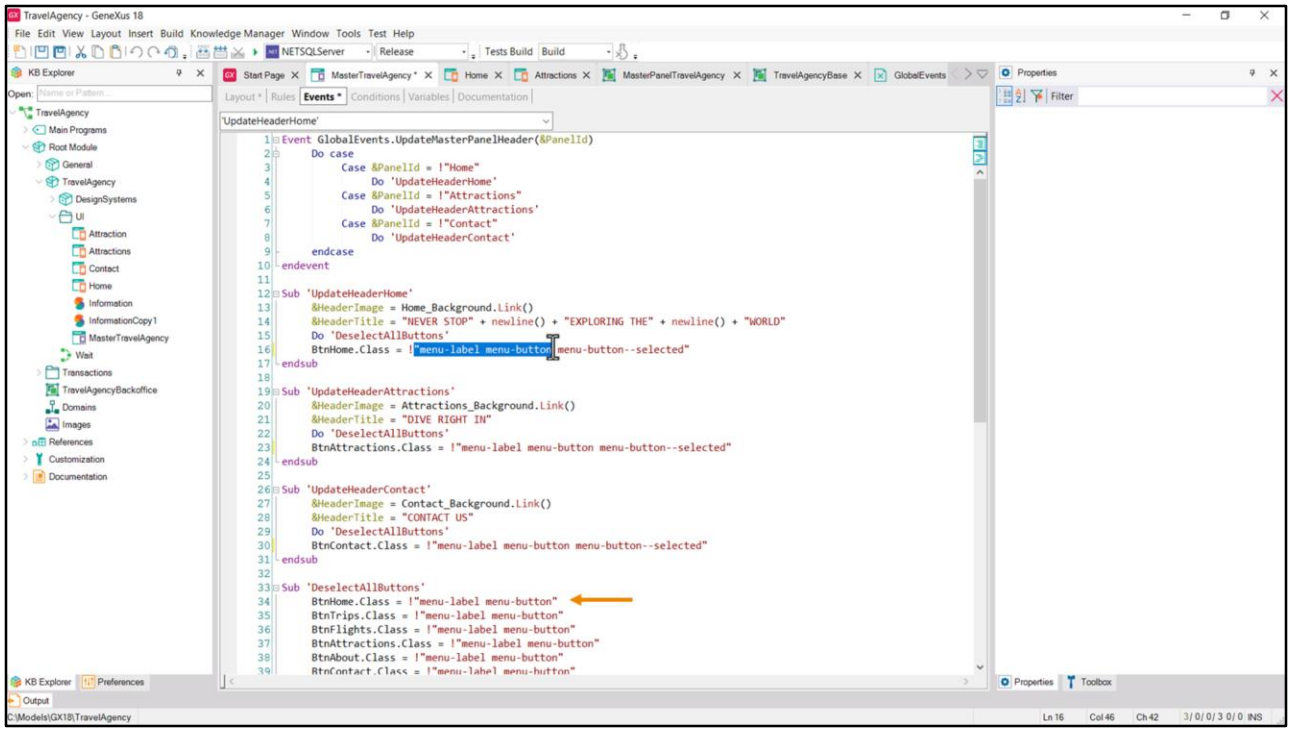
E sobrescreve-se nesta outra sub-rotina, para todos os botões suas classes, de modo que a do BtnHome, que tinha 3 classes, agora passa a ter 2. Mas, imediatamente, como é o panel Home que estamos executando, volta a recuperar as mesmas 3 classes.
Quando tudo isso termina, volta a renderizar esta parte do Header.

No total, neste processo, fizemos 2 renderizações do botão Home. A primeira com as classes estáticas, a segunda ao disparar o evento global. Apenas talvez para o caso de Home, isso passe mais despercebido, mas pensemos no que aconteceria se estivéssemos carregando o panel de Attractions, não o de Home. Na primeira renderização, ficará como selecionado o botão Home, e na segunda, quando for executado o evento global, lá o botão Home ficará desselecionado e agora passará a estar selecionado o de Attractions. Isso poderia causar um efeito de piscada entre a primeira renderização e a segunda.

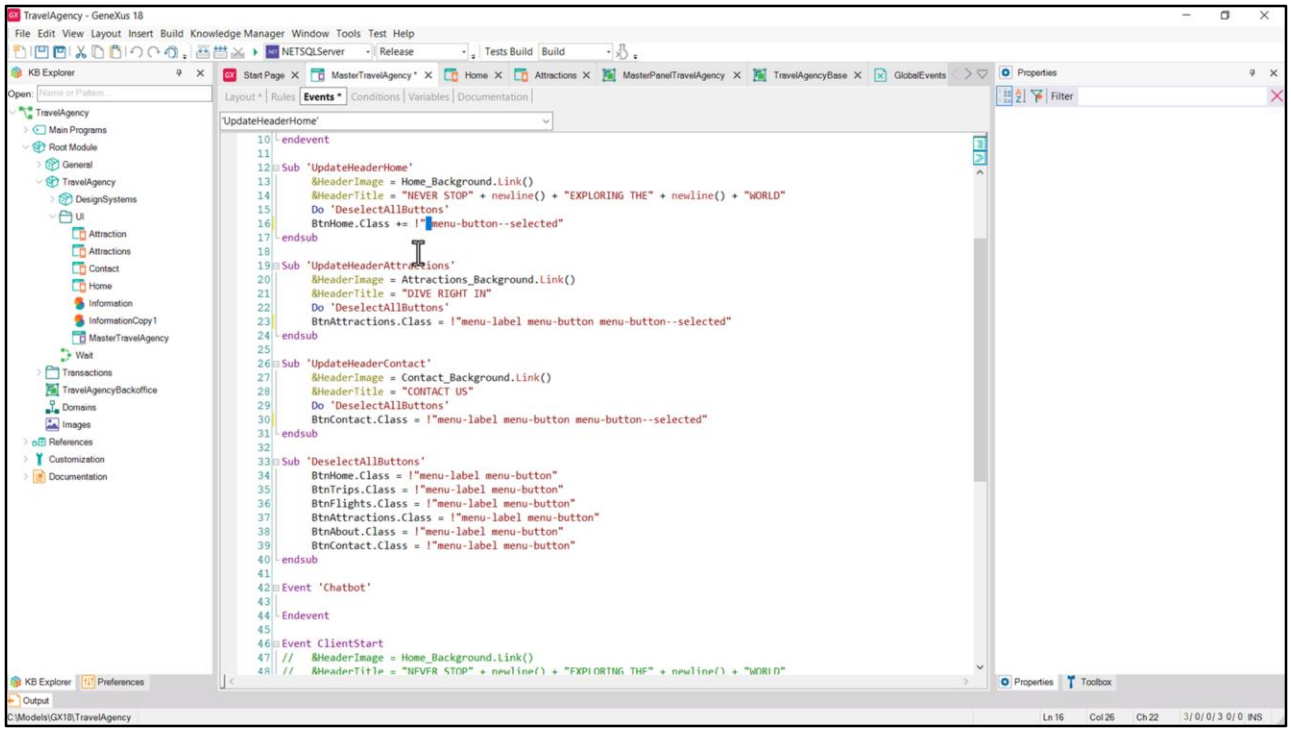
Na verdade, neste caso, será algo quase imperceptível, pois tudo é resolvido no cliente imediatamente.



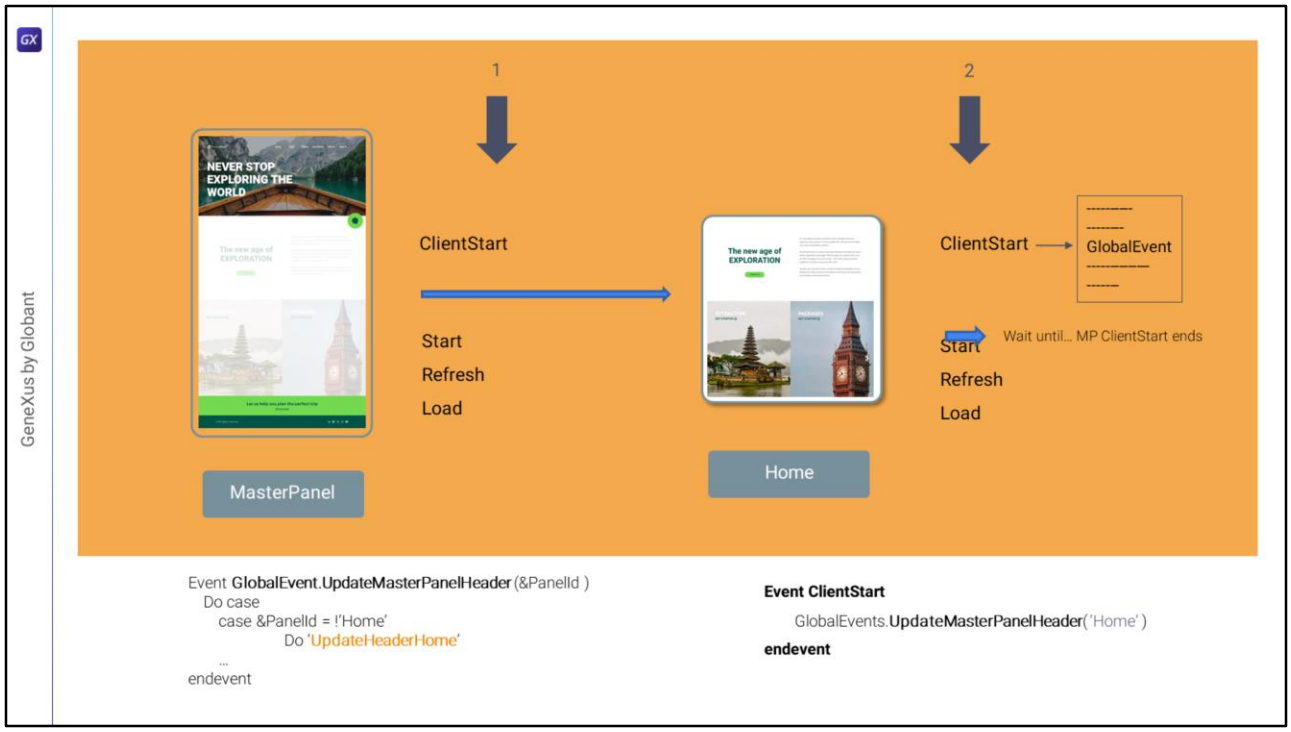
Mas, para fazer as coisas corretamente, removamos a atribuição estática à classe selecionada do botão Home. Assim, todos ficam uniformes, com as classes default, de modo que, quando forem sobrescritas com os mesmos valores na segunda renderização, não haverá chance de que se perceba qualquer piscada.



Me chamem de obsessivo, mas isso está me incomodando um pouco... por que volto a dizer que leva essas duas classes, se já está dito aqui!?



Vamos utilizar melhor o operador “mais igual” para que ao que já tinha, seja adicionado o que segue. Tenham cuidado de deixar um espaço em branco aqui, porque esta é uma concatenação de strings, e caso contrário, ficará colado isto a este outro. Faço a mesma alteração para os outros dois casos e executamos.



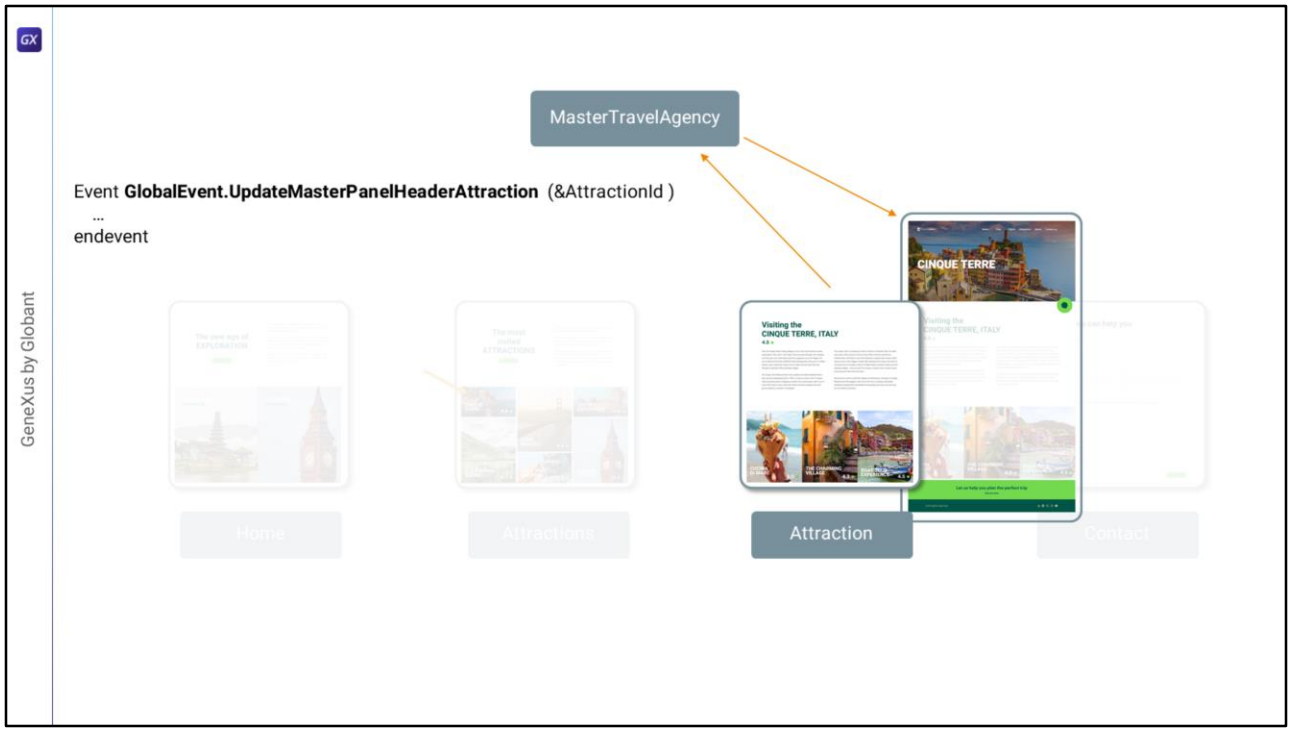
Resumindo então:

Ao executar nosso panel, começa-se pelo seu Master Panel, considerando os controles do layout de acordo com suas propriedades estáticas. Em seguida, é executado o evento ClientStart, que no nosso caso não possui nenhum comando assíncrono (na verdade, não tem nada programado), então é renderizado o Master Panel com o que se tem até aqui.

Dá-se a ordem de construir o panel para projetá-lo no ContentPlaceHolder, o que será feito nesta linha de execução concorrente, e paralelamente, na primeira, são executados os eventos Start, Refresh e Load, que estão vazios. Assim, nada acontece.

Na linha de execução do Panel, em sua construção, são consideradas as propriedades estáticas de seus controles, e é executado o evento ClientStart. Neste caso, temos o disparo de um evento global, que é escutado pelo Master Panel, então ocorrerá a execução concorrente deste evento e a continuação da execução do ClientStart do Panel, que não será interrompida pela outra. Quando termina, é renderizado o panel e, como o ClientStart do MasterPanel terminou há algum tempo (antes de iniciar a construção, na verdade), então seguirá com a execução de Start, Refresh, Load.
Paralelamente, o Master Panel, ao executar o evento global, como modifica conteúdos da User Interface, irá renderizá-los.

Como já comentei, está sendo trabalhado para que, ao navegar entre Panels com o mesmo MasterPanel, não seja reconstruído o MasterPanel, melhorando assim a performance e user experience.



Agora teríamos que ver o que ficou pendente: como fazer para que o Master Panel carregue no Header imagem e título da atração que recebeu por parâmetro o panel Attraction ao ser executado…

...e que o recebeu quando o usuário selecionou uma atração turística neste grid.

Sabemos que será a partir deste outro evento global. Para não me alongar mais aqui, faremos isso no próximo vídeo. Espero vocês.


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