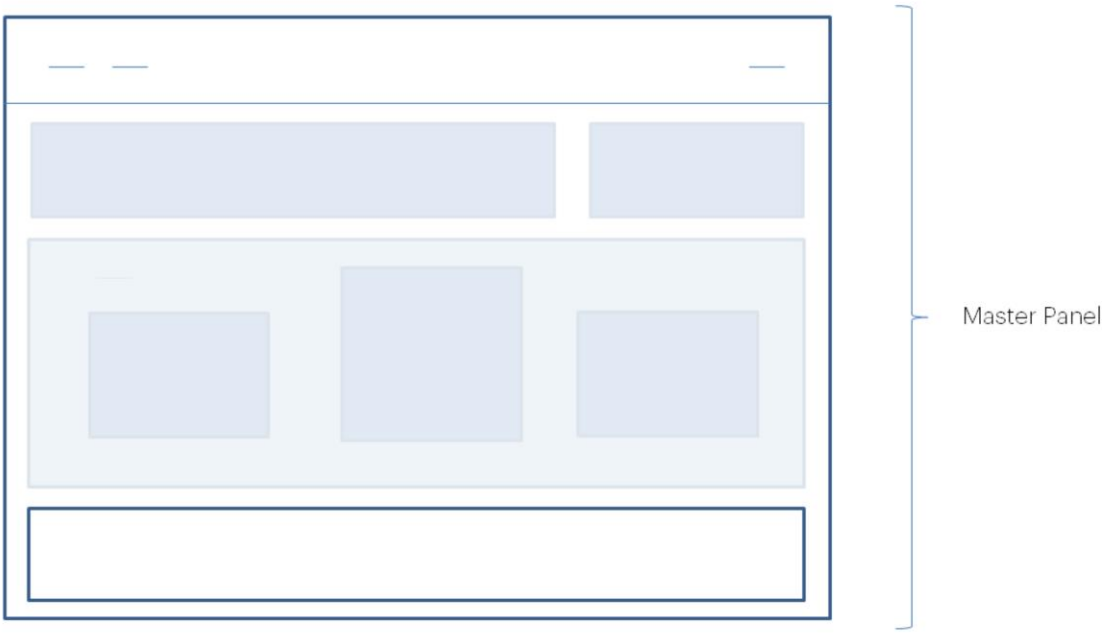
Eventos globais

Interação entre componentes de uma mesma tela

GeneXus

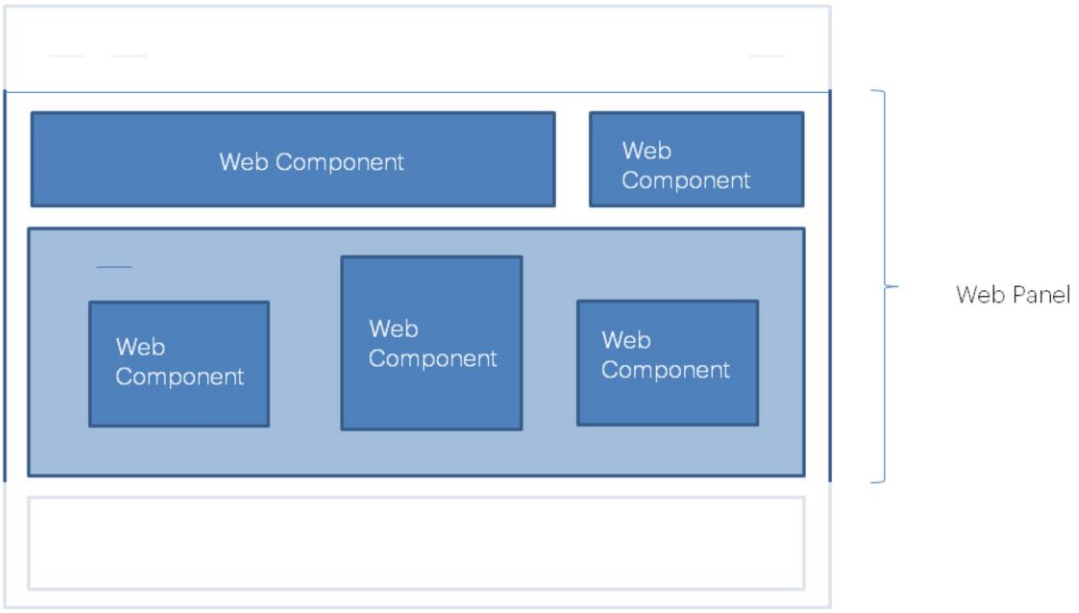
Screen composed by many screens

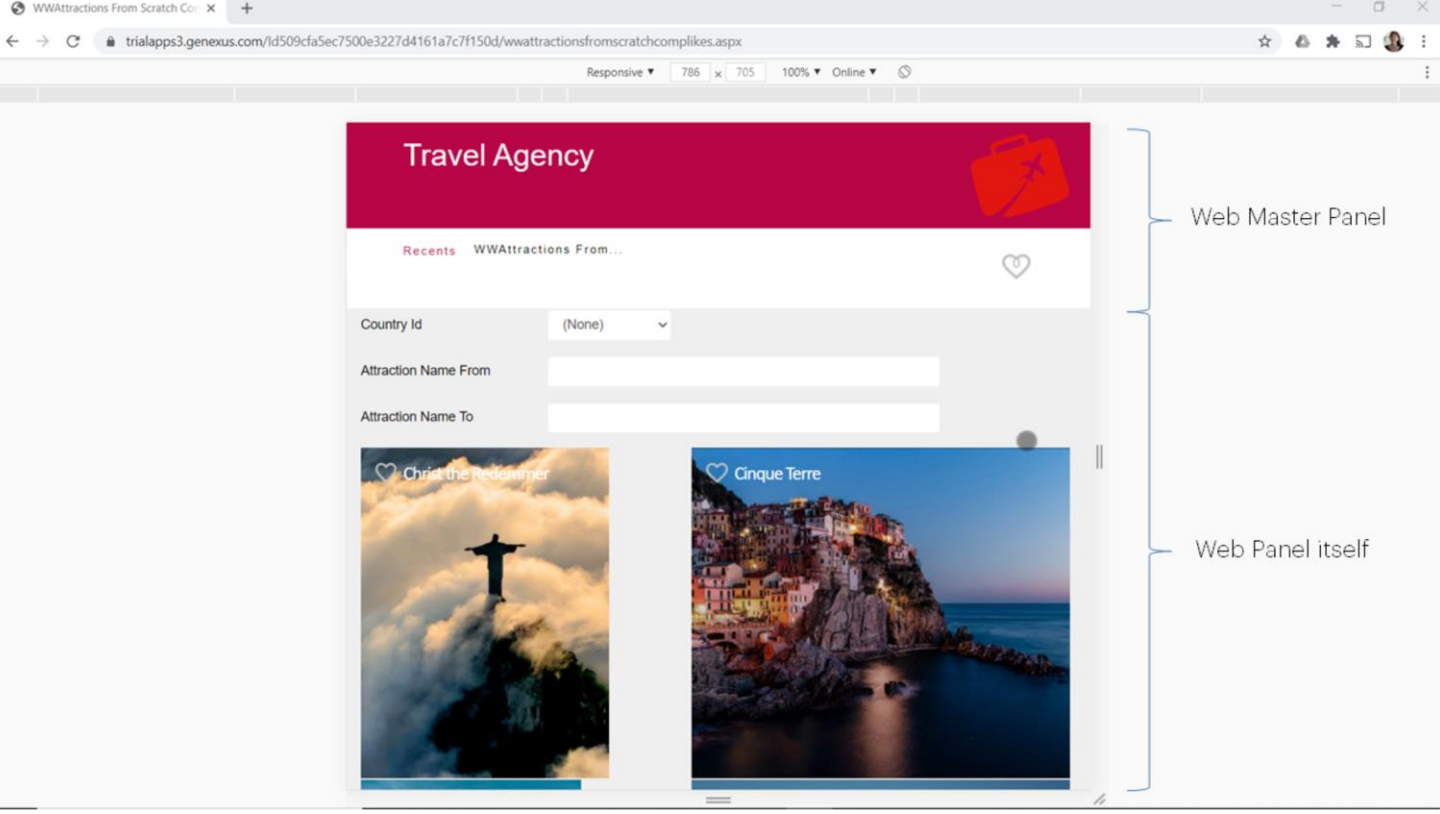


Como já sabemos, para reaproveitar, costumamos *componentizar* o máximo possível.

Assim, se observamos qualquer uma de nossas telas, em geral correspondem a vários objetos interagindo, e não a apenas um.

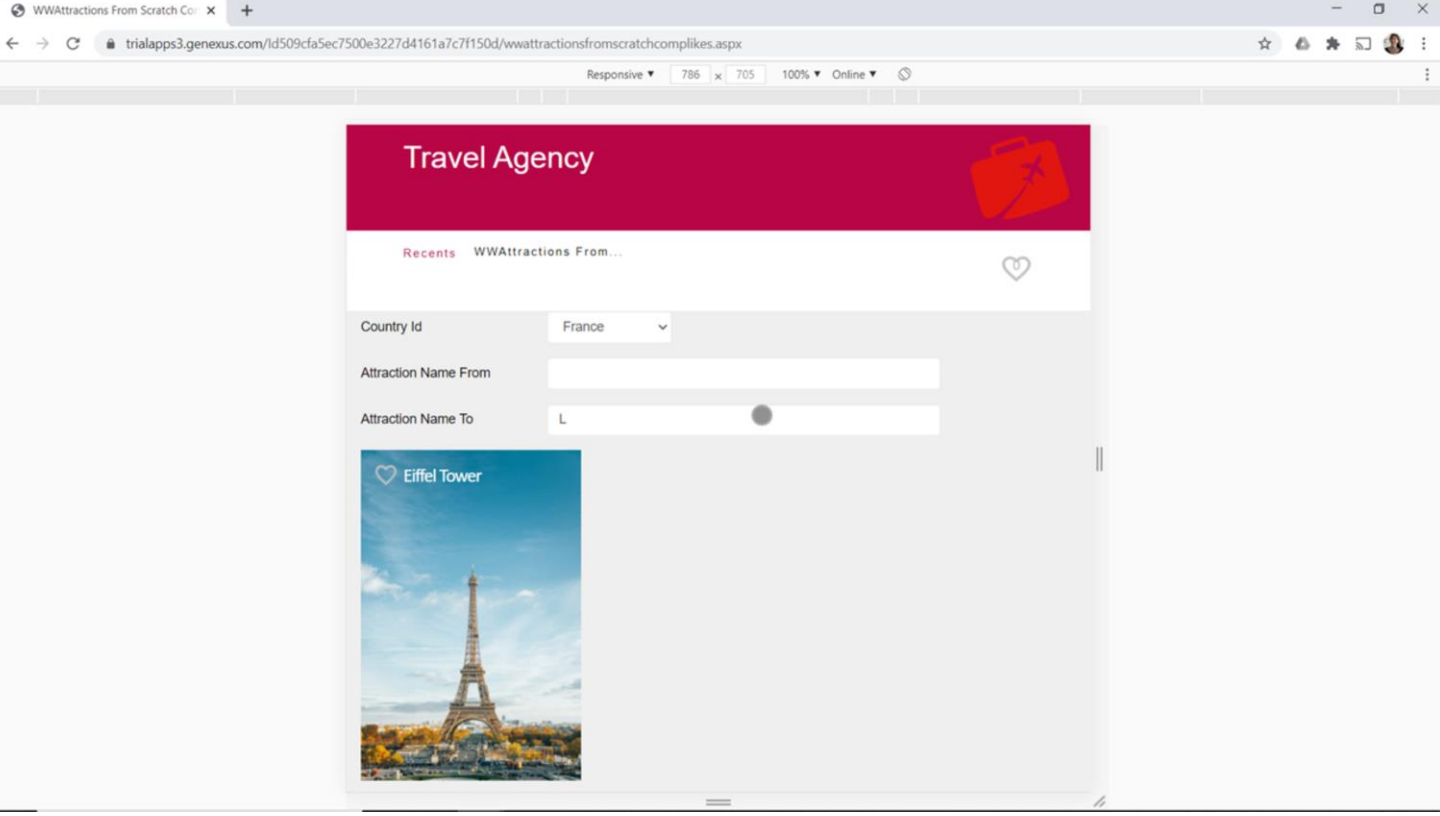
Screen composed by many screens



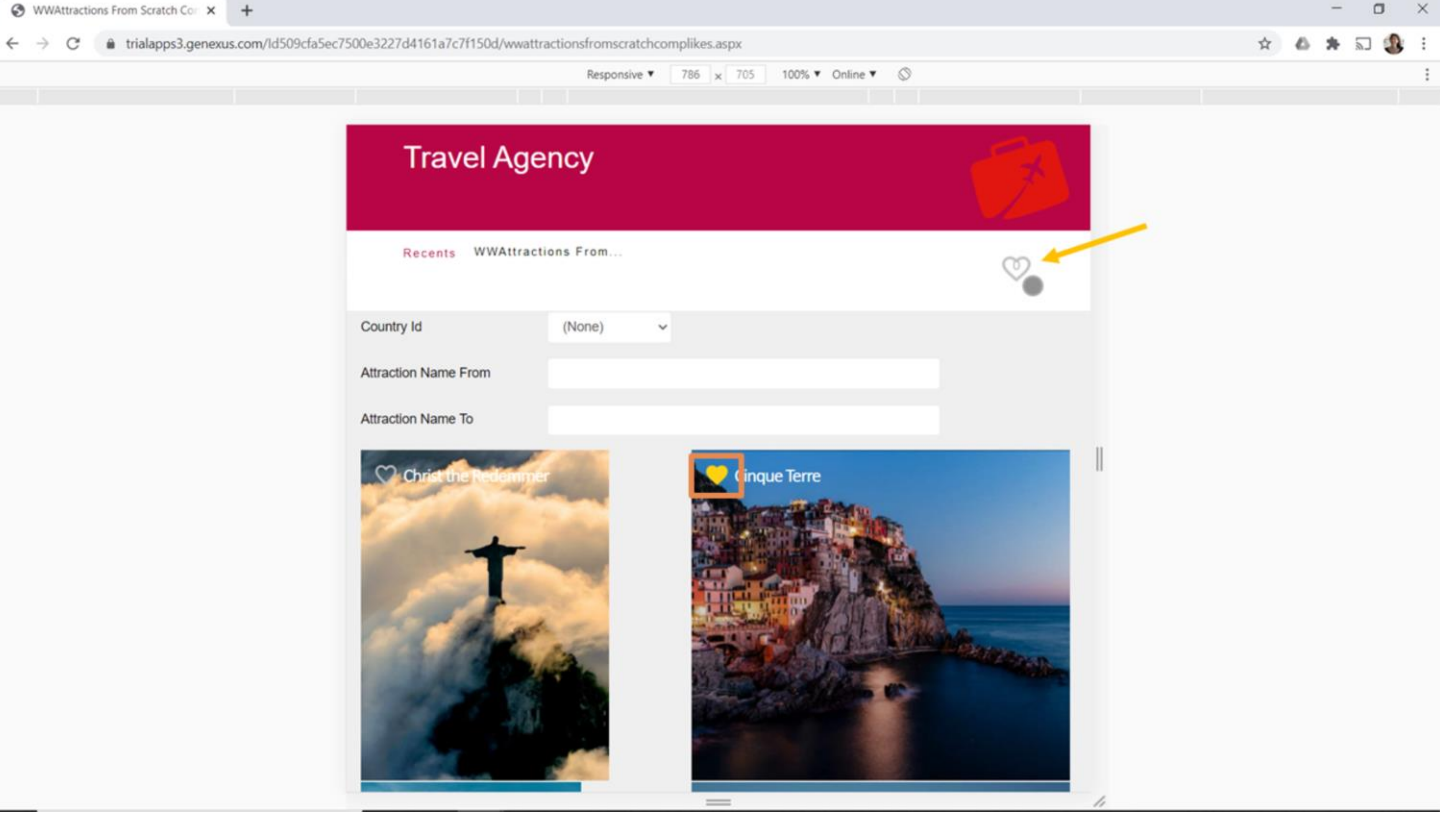


Por exemplo, é o caso de um web panel como o que vimos no curso inicial de GeneXus, onde se listavam as atrações turísticas que uma agência de viagens oferecia visitar. Não nos ocuparemos aqui do desenho, que está apenas esboçado e falta-lhe trabalho.

Estamos vendo uma tela não muito complexa, que terá toda uma parte que virá do panel mestre, e outra que é própria. Entre a própria, há partes implementadas, por sua vez, como componentes, ainda que em execução isto seja imperceptível.



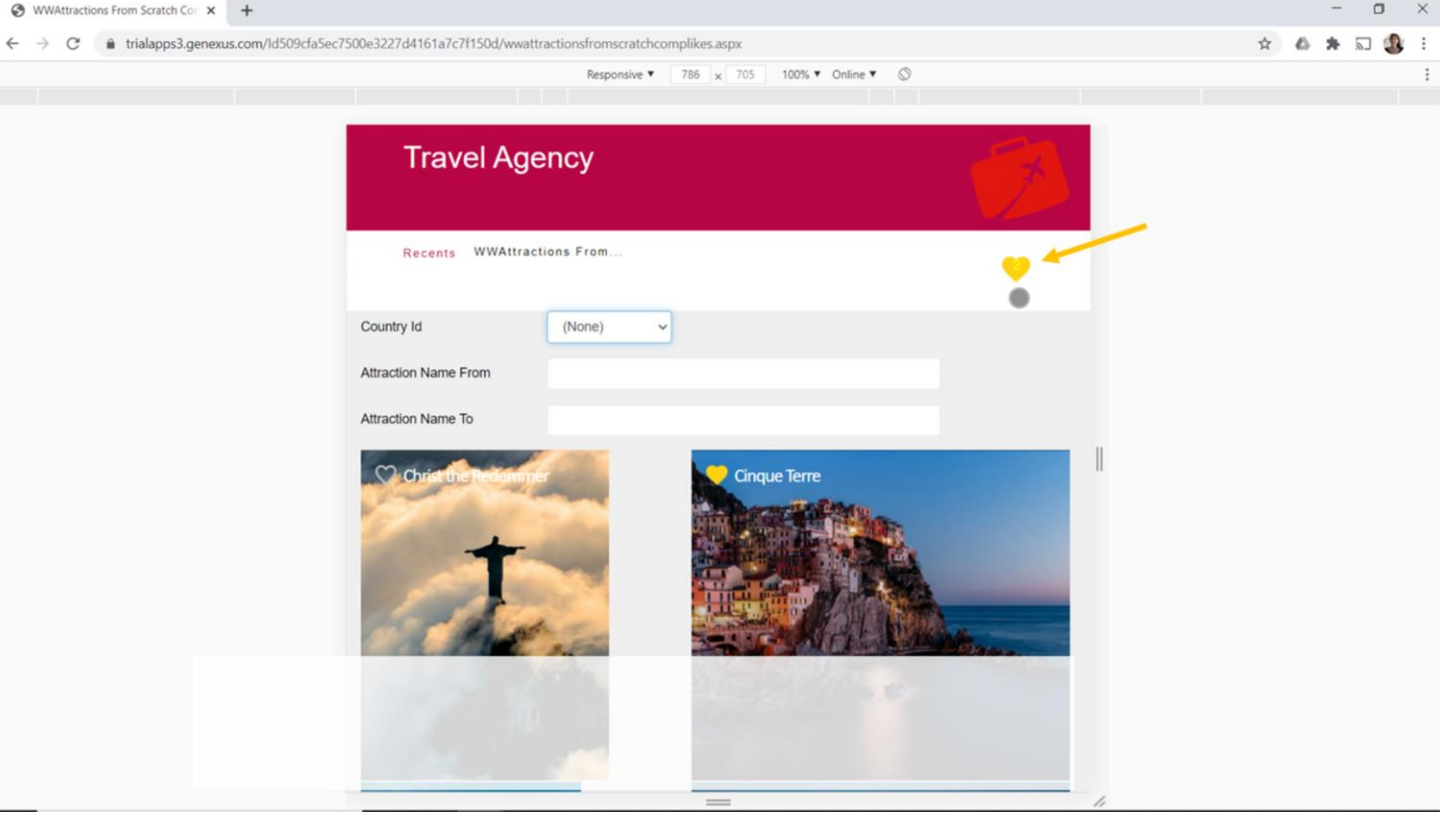
Podemos filtrar por país, e também por nome de atração.



Mas o mais interessante, que é o que nos permitirá introduzir o tema deste vídeo, é que vemos que temos agregada a possibilidade de *favoritar* atrações turísticas.

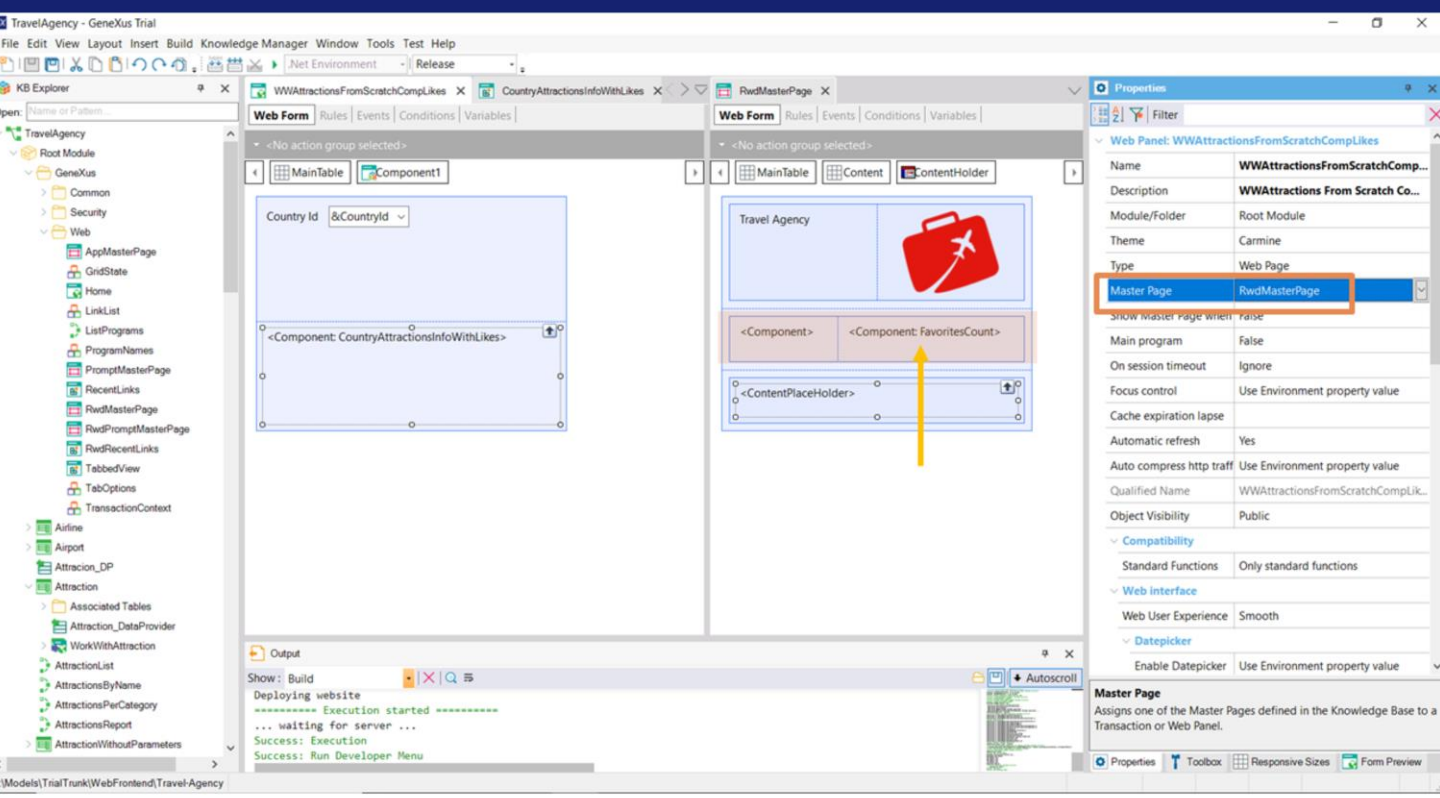
Por exemplo, marco esta e esta outra. Ou seja, marcamos duas atrações como favoritas.

No entanto, se vamos observar aqui em cima, está nos indicando um zero, como se não tivéssemos favoritas.



Atualizemos a tela. Agora sim vemos esse 2. Imaginaríamos, também, que se removo um favorito, este número se atualize, marcando 1.

A pergunta que vai nos guiar é como conseguimos que esse valor se atualize automaticamente conforme se *favorite* ou *desfavorite* cada atração.

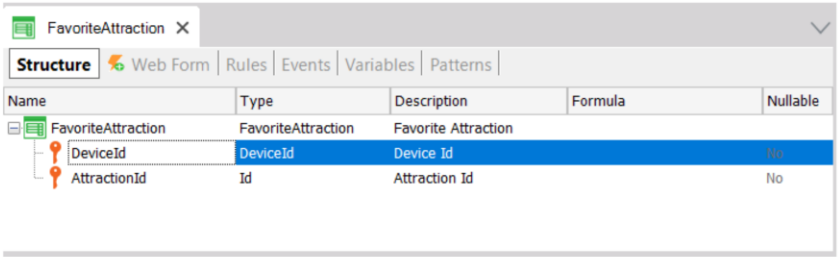


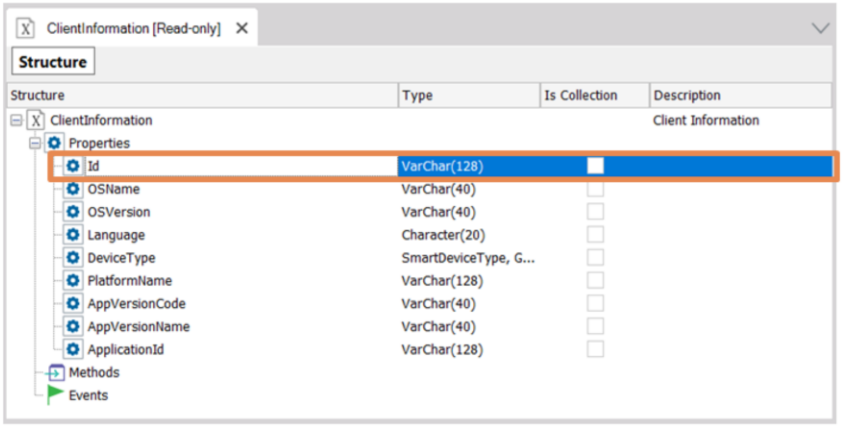
Vamos para GeneXus ver este Web panel.

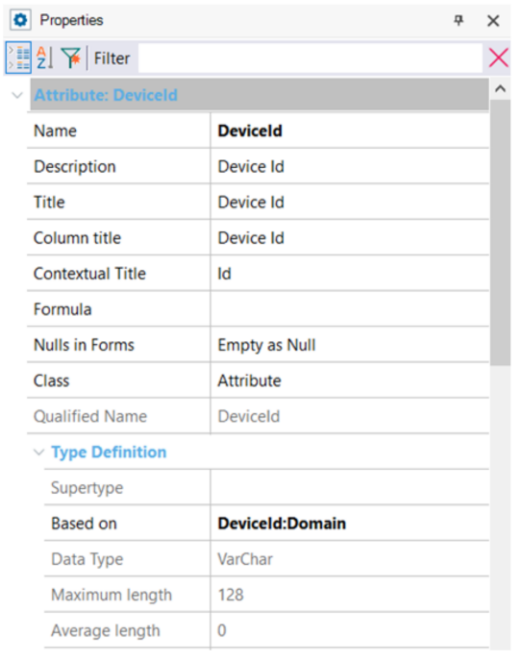
Vemos, antes de mais nada, que tem um panel mestre associado, que é o default. Então, o conteúdo da sua tela será carregado dentro do content place holder da página mestra.

E fora dele estará todo este outro.

Em particular temos dois componentes, um default, que é o que implementa os links recentes e o outro é um que adicionamos para mostrar a quantidade total de atrações turísticas que o usuário tem *favoritadas*.





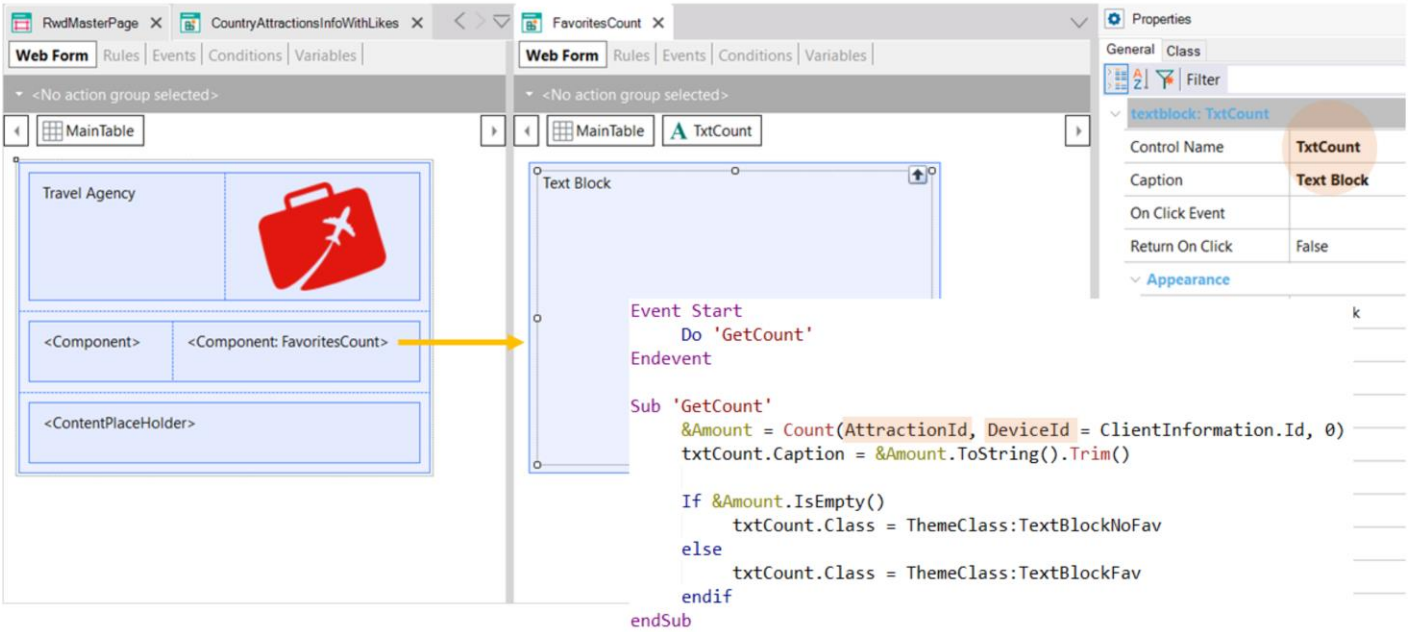


Para salvar as atrações que um usuário *favorita,* deveríamos ter uma tabela de usuários, para poder indicar atrações favoritas por usuário. Se não queremos, ainda, trabalhar com usuários (por exemplo, porque ainda não aplicamos GAM e estamos ainda resolvendo se vamos ter uma tabela própria de usuários ou não) então, podemos provisoriamente manter favoritos por instância de browser.

Para isso criamos esta transação com este atributo, ao qual especificamos este novo domínio que nós também criamos, com o mesmo tipo de dados que esta propriedade...

ClientInfomation é um external object do módulo GeneXus, cuja propriedade Id permite identificar o dispositivo cliente que está executando, tanto em forma nativa como web.

No caso de aplicações Web, a propriedade retorna um identificador do usuário, que persiste entre todas as sessões com o mesmo navegador e para a mesma aplicação. Então, o que fizemos foi adicionar uma transação com chave composta pelo dispositivo e a atração turística. Na tabela desta transação guardaremos os favoritos.

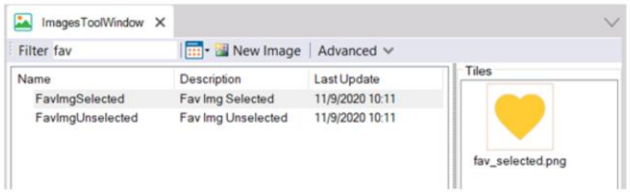


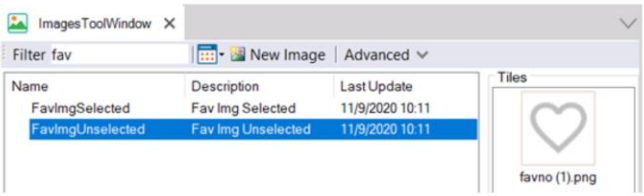
Se observamos o Web component que estamos carregando aqui, no Master Panel web, vemos que contém apenas um text block, deste nome, cujo caption carregamos em execução, a princípio apenas no evento Start, invocando esta sub-rotina, que a primeira coisa que faz é calcular a quantidade de atrações turísticas para este cliente, que se encontram na tabela que contém estes dois atributos (que é FavoriteAttraction).

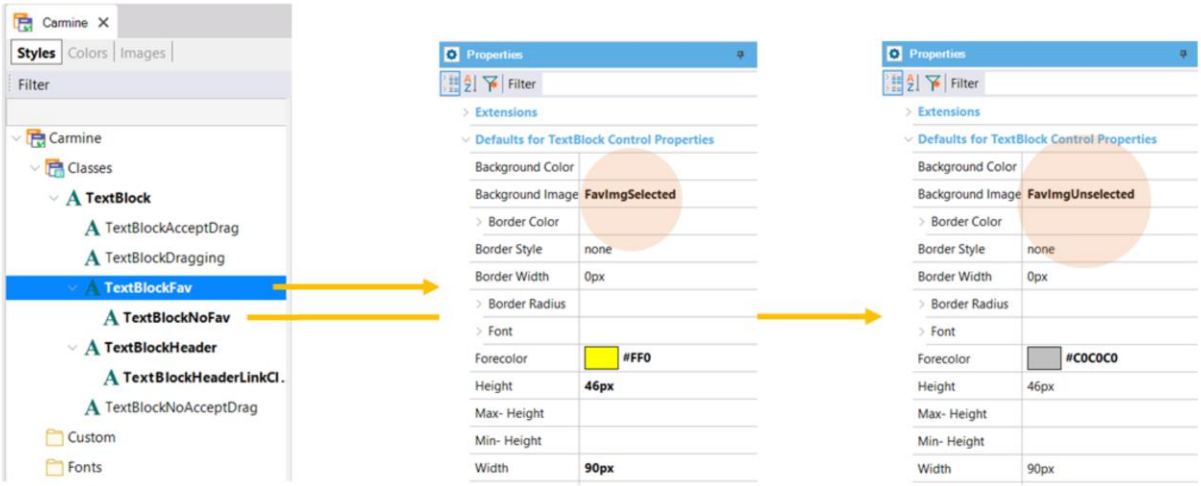
Aqui temos o valor que estamos procurando. Então, transformamos esse numérico em string, para atribuí-lo ao textblock que será exibido.

E para mostrá-lo com a imagem de favorito e na cor e formato adequados, adicionamos duas classes ao theme correspondente –que neste caso é Carmine-.

Se a quantidade de atrações é zero, atribuímos ao text block a classe que desenha o favorito vazio e, caso contrário, a que desenha o favorito amarelo cheio.

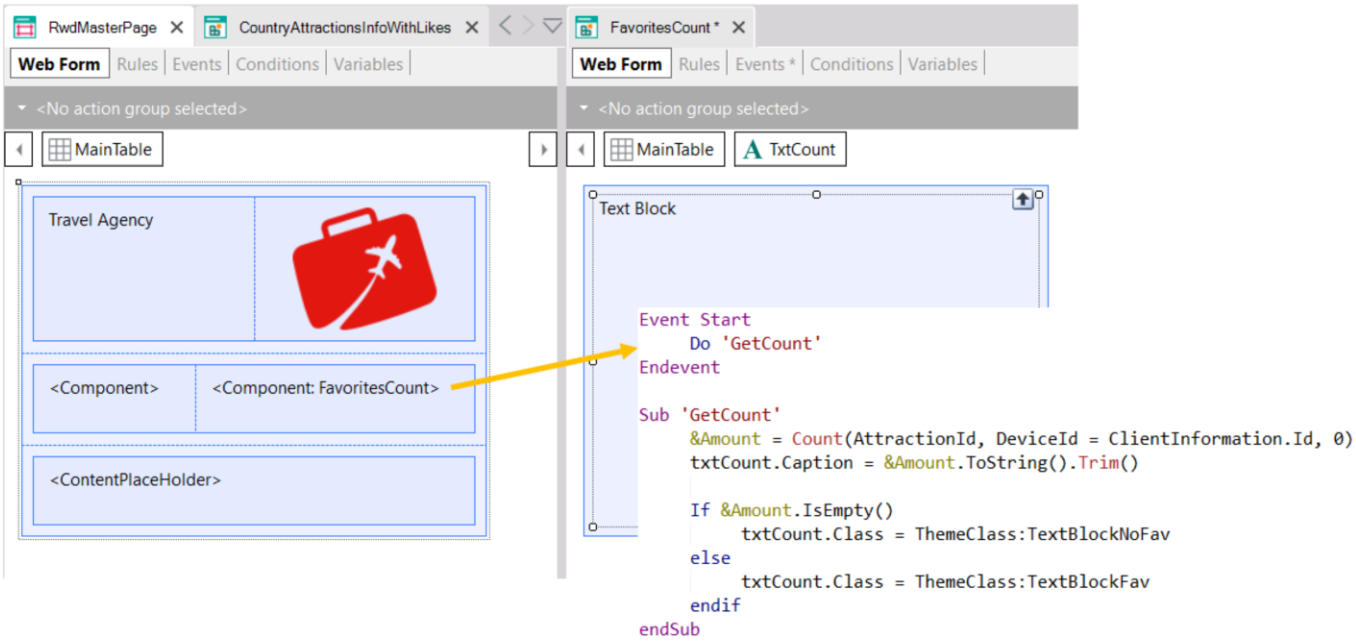






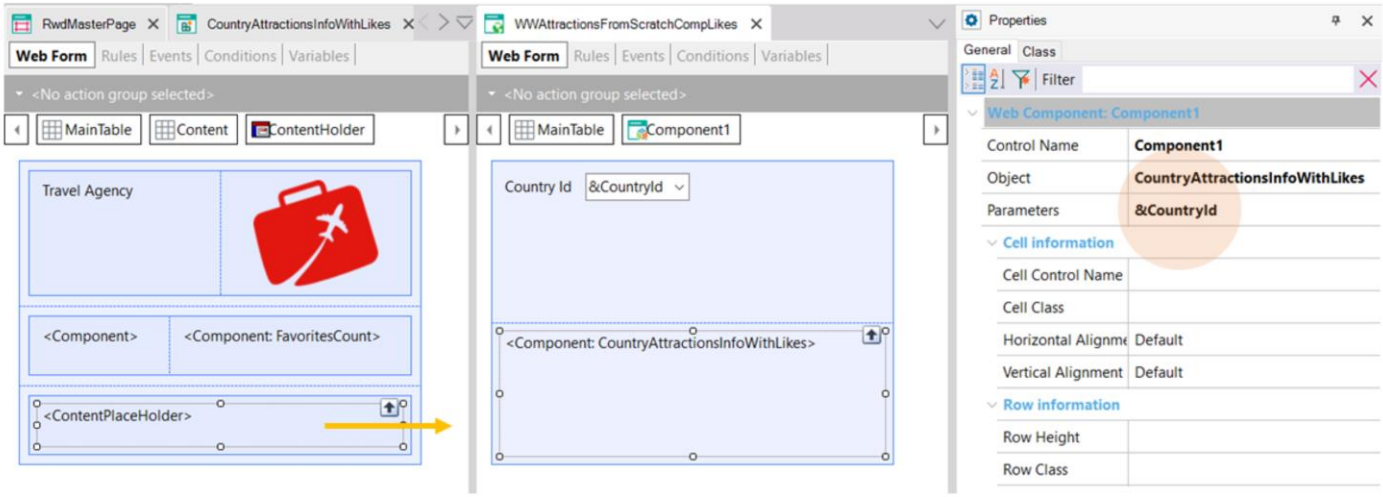
Para isto tivemos que inserir previamente duas imagens na KB...

Se olhamos as classes no theme, aqui indica-se na propriedade Background Image o coração amarelo cheio e, para esta outra classe, o cinza vazio.



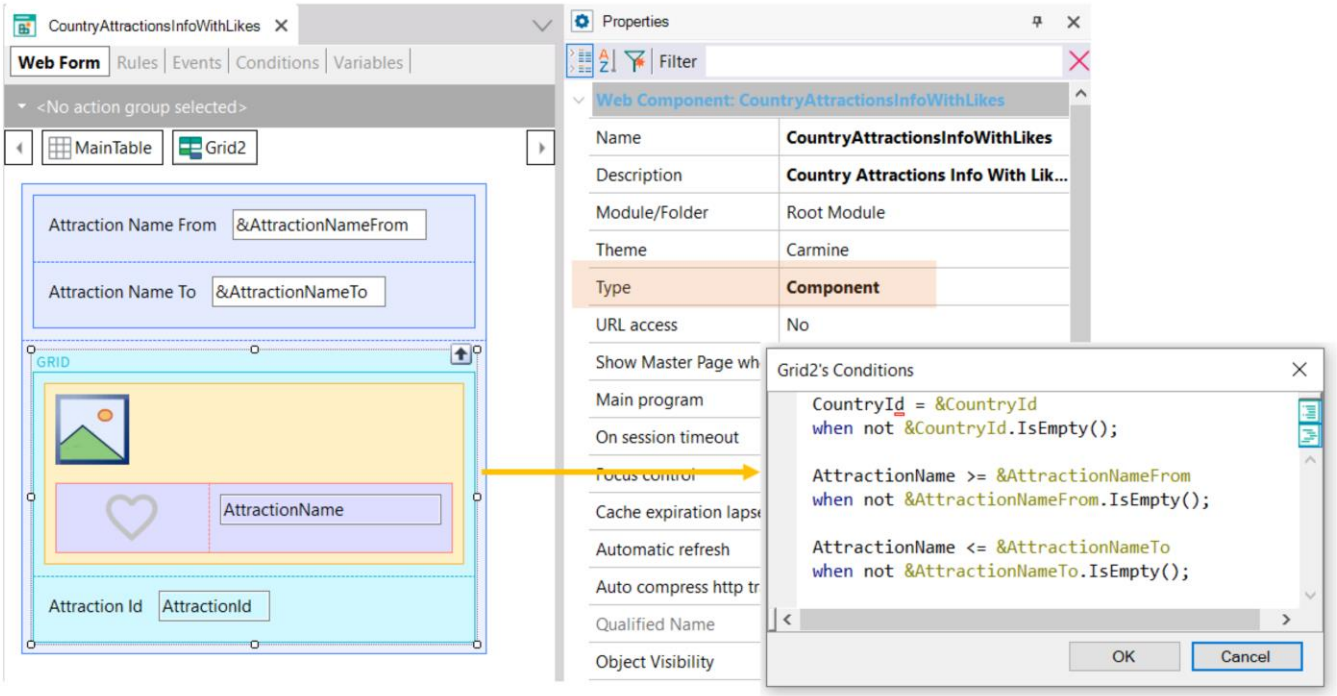
Então, este text block calcula a quantidade e a exibe com a imagem de fundo e as cores apropriadas.

Este evento Start será disparado quando o panel mestre for aberto.

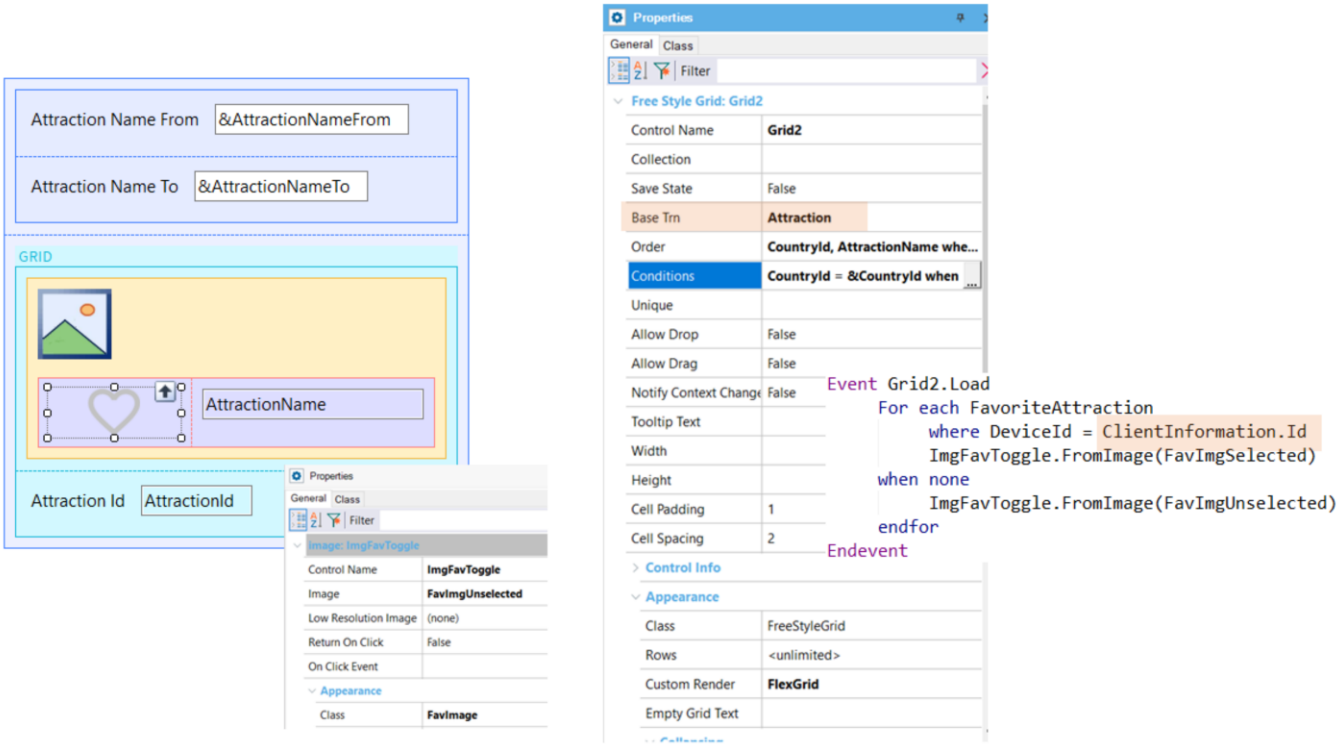



```
Event &CountryId.ControlValueChanged
    Component1.Object = CountryAttractionsInfoWithLikes.Create(&CountryId)
Endevent
```

Por outro lado, se observamos o que será carregado no ContentPlaceHolder para nosso Web panel... vemos que é uma tela com este combo box para escolher um país, e abaixo temos outro componente, ao que envia-se o valor desse país toda vez que o modifiquemos.

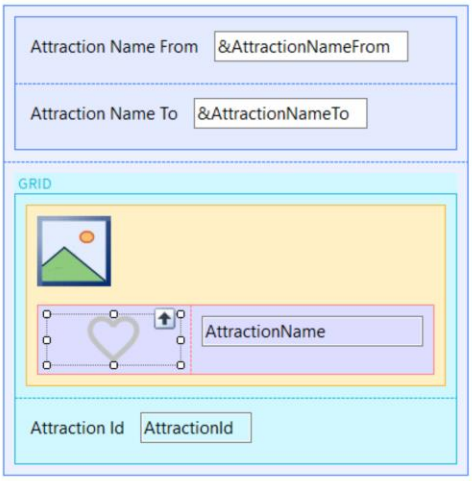


E se observamos quem se está carregando nesse web component, é este outro objeto web, de tipo componente, que tem um grid que filtra as atrações turísticas que mostrará, de acordo com o país recebido por parâmetro e os filtros da própria tela.



Escolhemos um grid Flex, que contém uma tabela Canvas, para poder sobrepor a foto da atração turística junto com seu nome e uma imagem para permitir ao usuário *favoritar* ou *desfavoritar* a atração.
Não nos deteremos aqui sobre o desenho de tudo isto.

Apenas observemos que no evento Load do grid, que tem como tabela base a Attraction, para cada atração turística que será carregada se executa este for each, que procura na tabela subordinada, FavoriteAttraction, se existe um registro para este dispositivo, o que está executando. Se existe, isso significa que a atração é favorita, por isso carrega a imagem com o coração cheio; e, caso contrário, com o coração vazio.




```
Event ImgFavToggle.Click
     &isFavorite = ToggleFavorite(AttractionId)
     If &isFavorite
          ImgFavToggle.FromImage(FavImgSelected)
     else
          ImgFavToggle.FromImage(FavImgUnselected)
     endif
Endevent
```

ToggleFavorite

Source | Layout | Rules | Conditions | Variables

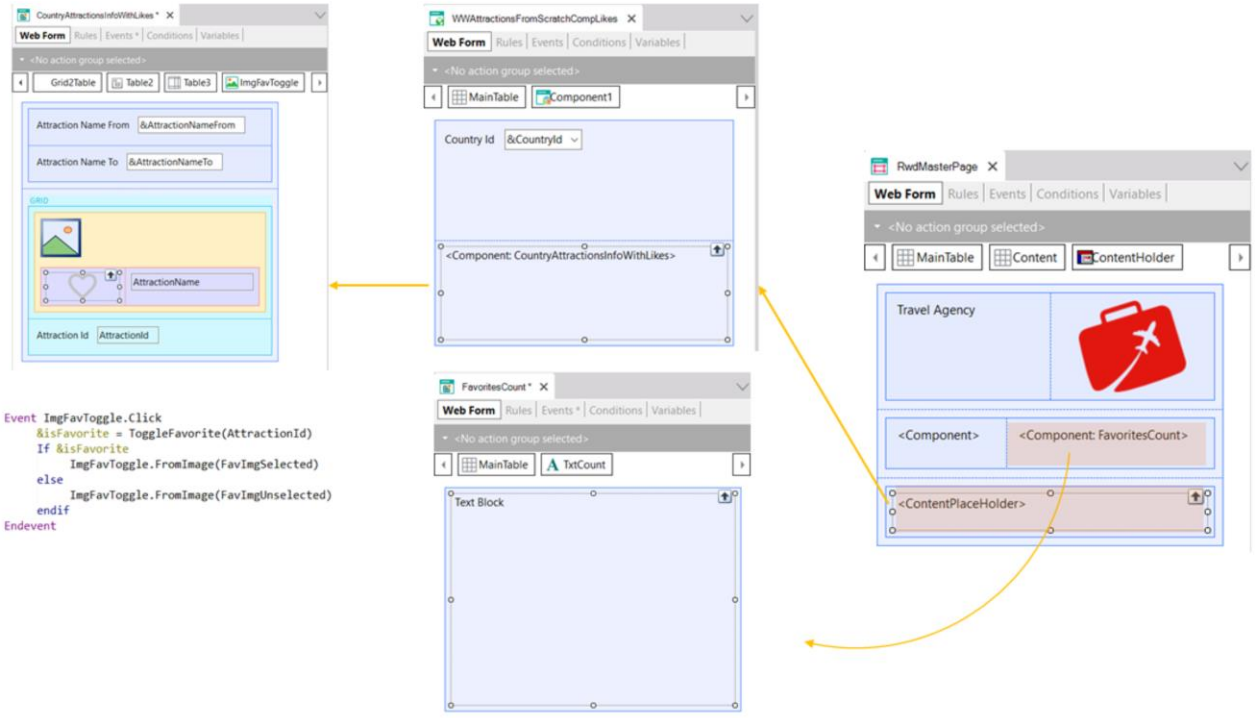
Subroutines

```
1  For each FavoriteAttraction
2       where DeviceId = ClientInformation.Id
3       where AttractionId = &AttractionId
4       &isFavorite = False
5       Delete
6  when none
7       &isFavorite = True
8       new
9            DeviceId  = ClientInformation.Id
10           AttractionId = &AttractionId
11      endnew
12 endfor
```

Mas, de tudo isto, o que realmente nos importa é outra coisa: a programação do evento click sobre esta imagem.

Uma vez carregado o grid de atrações, quando o usuário clica no favorito de uma atração, chamamos um procedimento que determina se a atração já estava *favoritada*, neste caso, a exclui da tabela e indica a operação para fazer o switch da imagem pela vazia. E se for ao contrário, faz o oposto, insere.

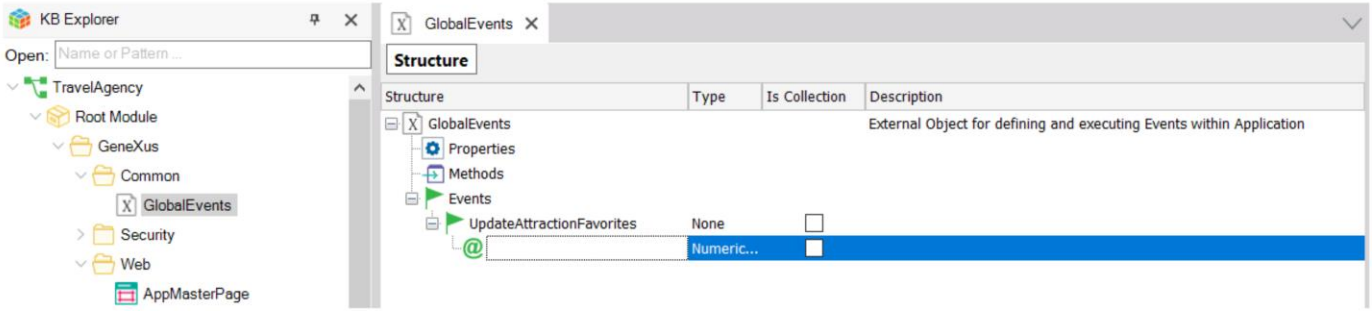
Agora, ao fazer isto, o total *favoritado* deveria atualizar-se, mas não está fazendo isso.



Recapitulando... este evento se dispara em um web component que está dentro deste web panel, que por sua vez é executado dentro do Web Master Panel, este, que é onde se cria também o componente que queremos se atualize.

Em outras palavras: queremos que um evento de usuário de um componente carregado neste panel permita realizar uma ação sobre um componente com o qual não tem outra relação além de encontrarem-se indiretamente reunidos na mesma tela.

Global Events: default external object



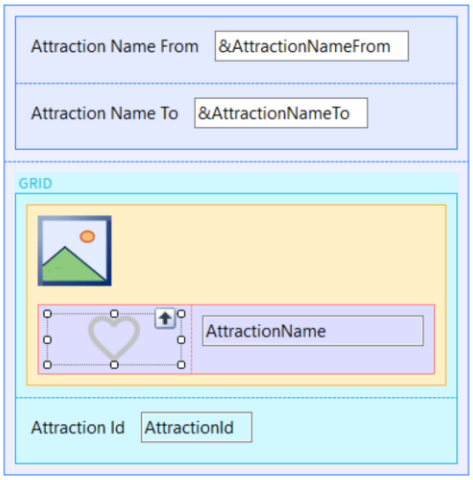
Para permitir a interação, toda KB virá com este objeto predefinido, Global Events, para que possamos modificá-lo.

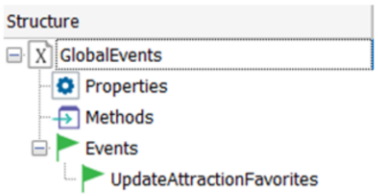
Poderíamos salvá-lo com outro nome e criar tantos objetos particulares quanto queremos.

Em nosso caso, vamos modificar o predefinido, que vem vazio, adicionando-lhe um evento ao qual chamaremos assim.

Os eventos adicionados poderão manejar parâmetros, se forem necessários para a comunicação entre componentes.

Em nosso caso só precisamos que um saiba que outro o disparou. Não precisamos de parâmetros.



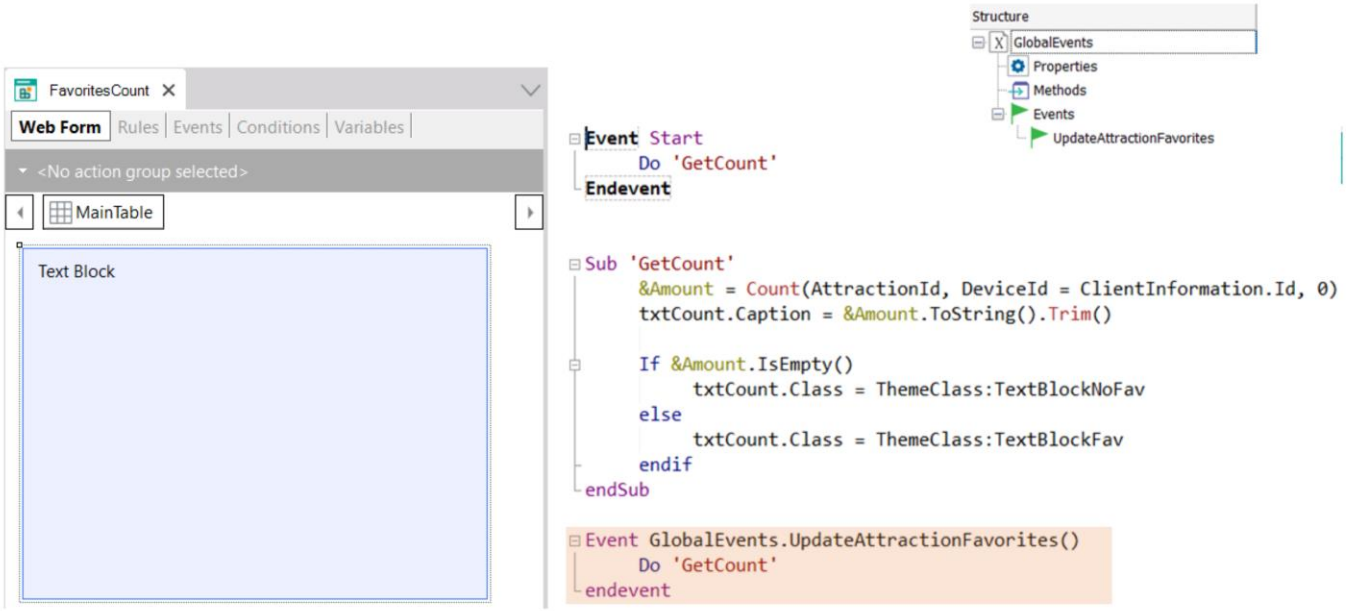



```
Event ImgFavToggle.Click

    &isFavorite = ToggleFavorite(AttractionId)
    If &isFavorite
        ImgFavToggle.FromImage(FavImgSelected)
    else
        ImgFavToggle.FromImage(FavImgUnselected)
    endif
    GlobalEvents.UpdateAttractionFavorites()
Endevent
```

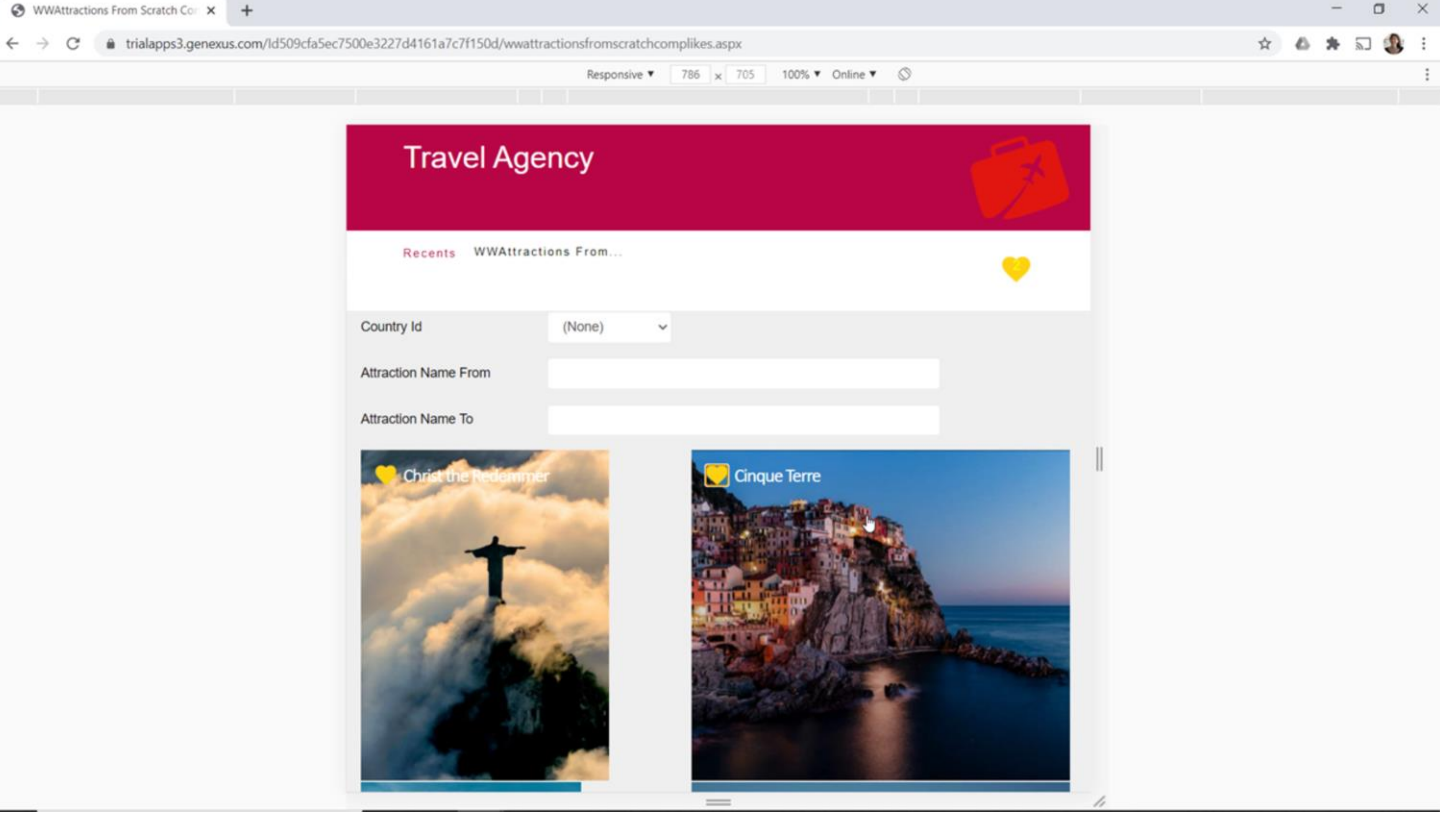
Tendo esta forma global de comunicação, a colocamos para funcionar.

Vamos para o panel que deve disparar o evento. O disparo será cada vez que se clique sobre a imagem. Nesse momento invocamos o objeto externo GlobalEvents, evento: o único que há no momento. Com isso está sendo reportado o disparo do evento.

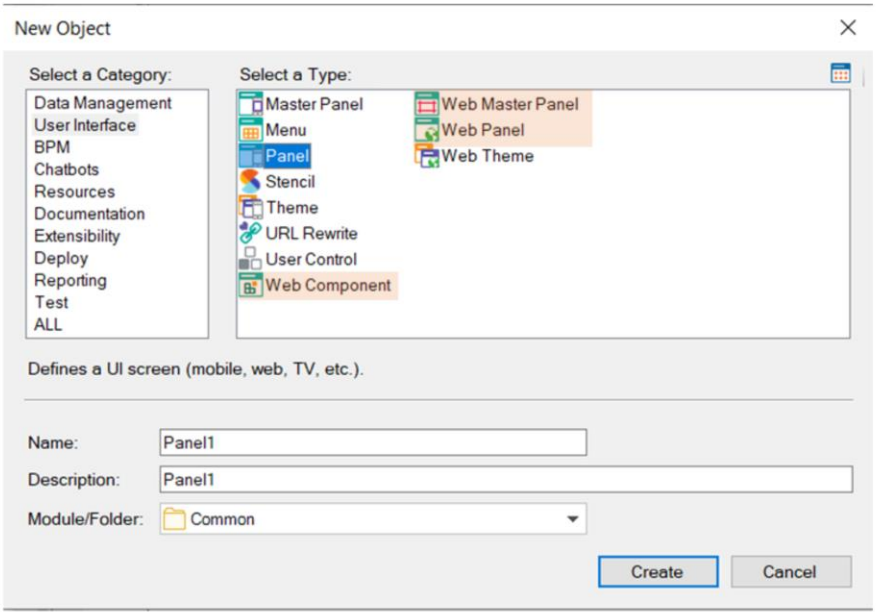


Em seguida, o que temos que fazer é que o componente que conta as atrações para mostrá-las se subscreva a este evento, para poder escutá-lo toda vez que ocorra na tela na qual quem escuta se encontra.

Para isso simplesmente declaramos escutar o evento. E ali programamos o que queremos que se execute quando o evento ocorre. Neste caso é voltar a calcular a quantidade de atrações.



Executemos para testar. Desselecionamos e vemos que se está atualizando. Agora marcamos, marcamos... A comunicação está sendo efetuada com êxito.



Embora aqui vimos um exemplo de comunicação entre painéis web com componentes, o mesmo vale para comunicar painéis multi-experience (por exemplo, para aplicações nativas) com componentes.

Para aprender mais sobre este tema, pode dirigir-se à nossa wiki.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications