Telas web com foco em Customer-facing

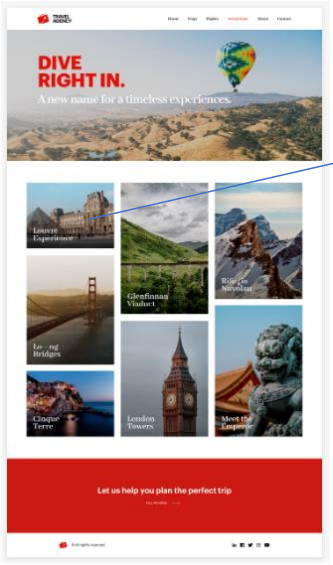
Primeiros passos com Angular

GeneXus™

Até agora vimos como construir telas com foco no back-office da aplicação, ou seja, na parte que usam os funcionários da agência de viagens para inserir e manter os dados da empresa. Agora veremos como desenhar e implementar as telas que utilizarão os clientes da agência para consultar informações, o que constitui a parte customer-facing da aplicação.

Requisito da Agência: Aplicação Customer facing

Objetos Panel

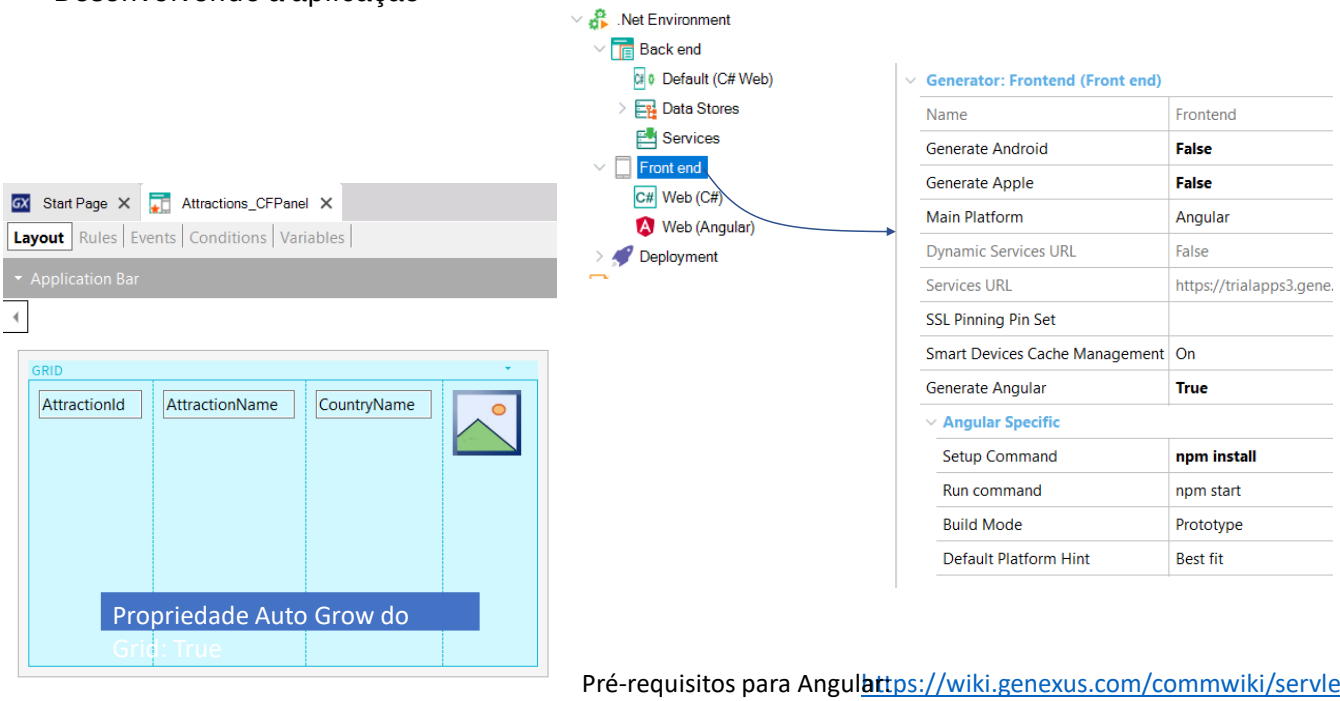


A agência de viagens nos pede para construir para seus clientes, uma tela web que mostre as atrações turísticas disponíveis e poder interagir com a informação, por exemplo, filtrando os dados na tela para refinar uma busca ou ver o detalhe de uma atração selecionada.

Antes vimos como construir telas com web panels, veremos agora como implementá-las com o objeto panel, o que nos permitirá gerar a aplicação em Angular.

Recordemos que se quiséssemos, poderíamos usar estes mesmos objetos panel, para gerar as telas de nossa aplicação móvel, para dispositivos Android ou Apple.

Desenvolvendo a aplicação



Pré-requisitos para Angular https://wiki.genexus.com/commwiki/servlet/wiki?

Vamos criar a lista de atrações disponíveis, de forma a poder, em seguida, clicar em uma que nos interesse e ver mais informações dessa atração. A ideia é construir um panel semelhante ao webpanel WWAttractionsFromScratch que vimos anteriormente.

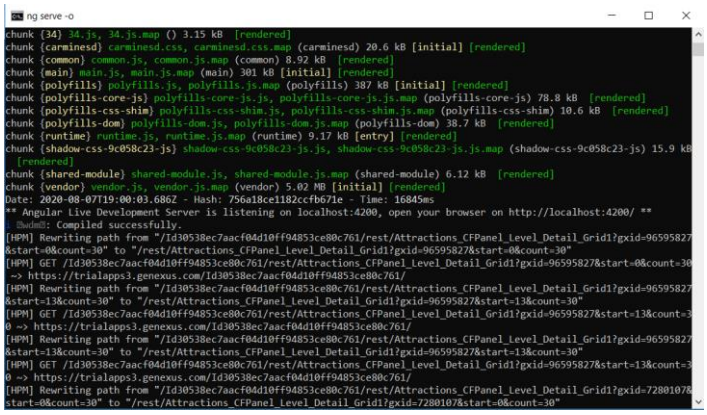
Criamos um folder FrontendAngular e criamos um objeto do tipo Panel de nome Attractions_CFPanel. Como vemos, aqui também temos um formulário onde podemos arrastar controles a partir da barra de ferramentas.
Começamos arrastando um Grid e selecionamos os atributos AttractionId, AttractionName, CountryName e AttractionPhoto. Depois, para Attraction Id colocamos a propriedade Visible em False e na propriedade Base Trn do grid, atribuímos a transação Attraction.

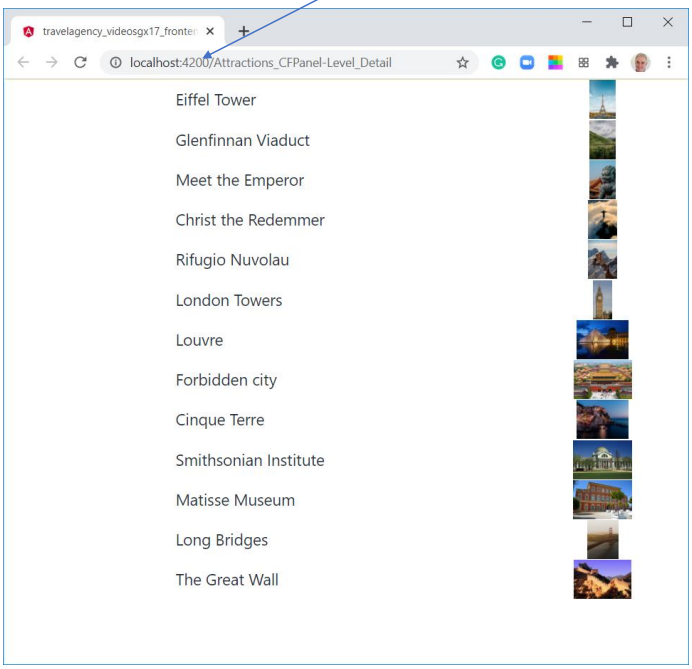
Agora atribuiremos Angular como gerador. No KB Explorer clicamos em Frontend e observamos que há uma propriedade chamada Generate Angular que seu valor por padrão é False, então a colocamos em True. Também vemos que Generate Android e Generate Apple estão por padrão em True, mas como não nos interessa, por enquanto, gerar para dispositivos móveis, as colocamos em False.

Para poder gerar em Angular, devemos instalar o software mencionado no artigo da wiki, mostrado na tela.

Execução da aplicação

O valor do porto pode mudar e será atribuído

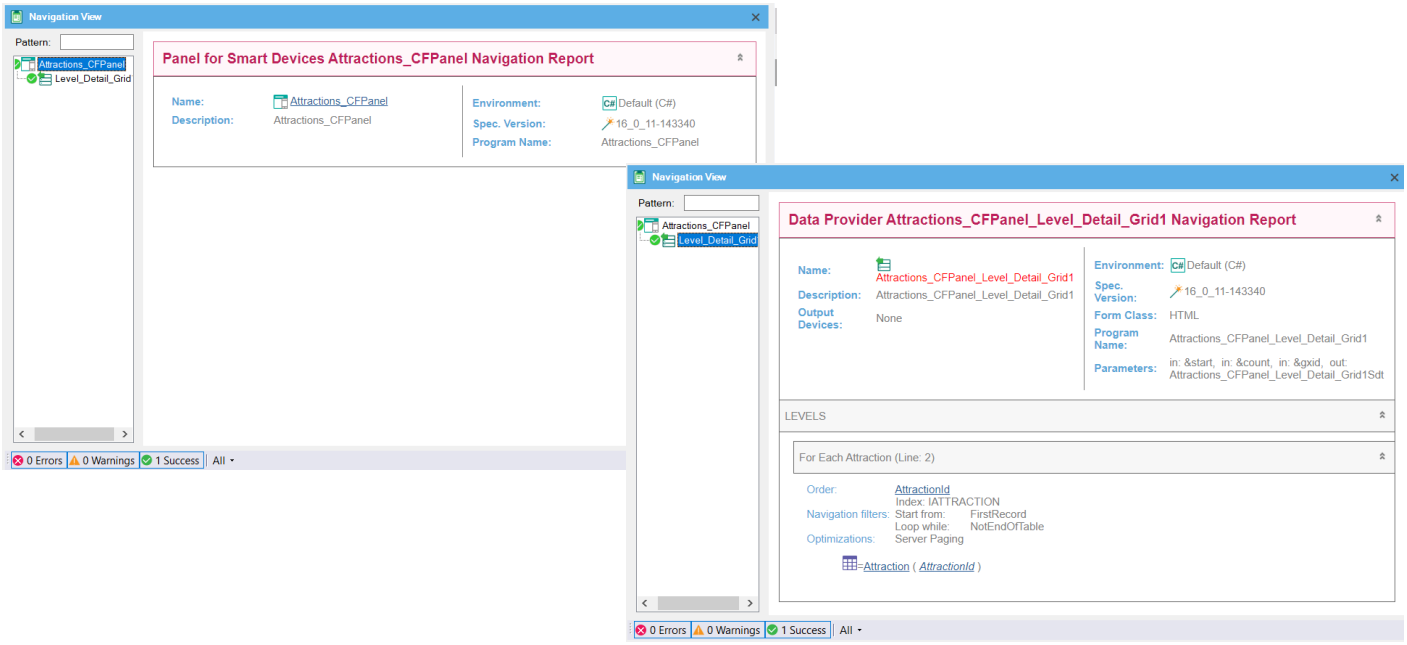




Para executar, primeiro colocamos o panel que acabamos de criar como Main, clicamos com o botão direito sobre ele e selecionamos Run. Vemos que se abre uma janela de linha de comandos que mostra a execução do servidor e, depois de alguns minutos, abre o navegador com a aplicação em execução.

Obviamente o desenho deixa bastante a desejar, mas já nos ocuparemos disso. A aplicação está sendo executada por padrão na url : http:// localhost:4200, que corresponde ao endereço do servidor local para desenvolvimento, onde a aplicação foi instalada automaticamente.
Já temos funcionando nossa primeira aplicação em Angular.

Listas de navegação



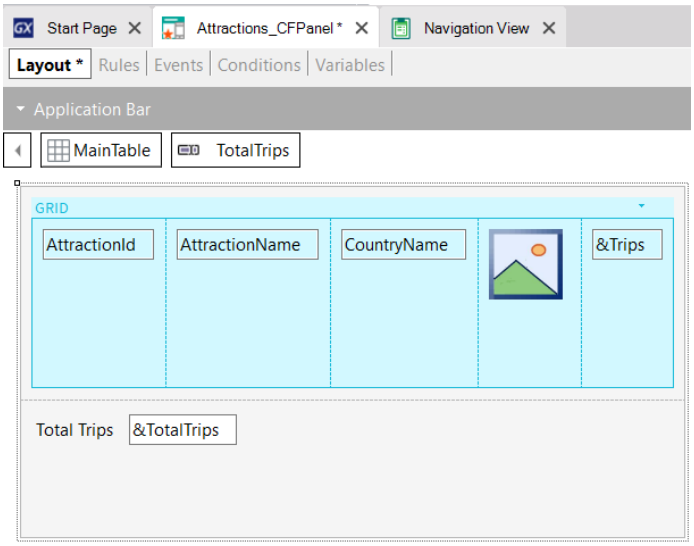
Se formos à lista de navegação do objeto panel Attractions_CFPanel vemos que aparecem duas entradas. Se selecionamos a que tem o nome do panel, a lista está vazia e não aparece nenhuma referência à tabela Attraction. Isso está no nó chamado Level_Detail_Grid1, onde vemos que a informação é similar à que veríamos com um web panel com tabela base Attraction, já que o grid do panel está percorrendo a tabela Attraction para mostrar as atrações turísticas.

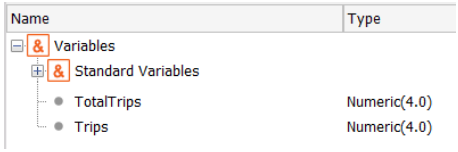
O nó da lista de nomes Attractions_CFPanel corresponde ao panel mesmo, e inclui informações de elementos de User Interface, como, por exemplo, a navegação dos Dynamic combo. Como em nosso panel não temos nenhum elemento na parte fixa, a lista aparece vazia.

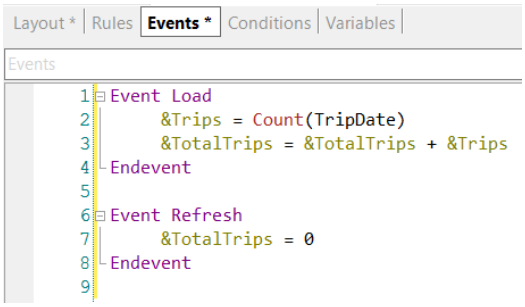
O nó chamado Level_Detail_Grid1 corresponde ao grid que inserimos. Vemos que se percorre a tabela base Attraction tal como esperávamos, ao ter inserido atributos de Attraction no grid. No título do relatório diz que a navegação corresponde ao Data Provider Attractions_CFPanel_Level_Detail_Grid1, que é o data provider publicado como serviço no back-end e invocado pelo panel para acessar a base de dados e recuperar as atrações.

Aqui temos a prova do que vimos antes, que a arquitetura destas aplicações customer-facing com Angular, têm a lógica a nível do cliente e se invocam serviços no servidor para acessar a informação da base de dados.

Adicionamos total de viagens de cada atração e total global de viagens







Para que nosso panel fique mais parecido com o webpanel de Trabalhar com Atrações visto antes, vamos adicionar mais uma coluna ao grid com o total de viagens de cada atração e fora do grid, o total de viagens de todas as atrações. Criamos as variáveis &Trips e &TotalTrips, adicionamos a &Trips no grid colocando sua propriedade Label position em None e a &TotalTrips abaixo do grid.

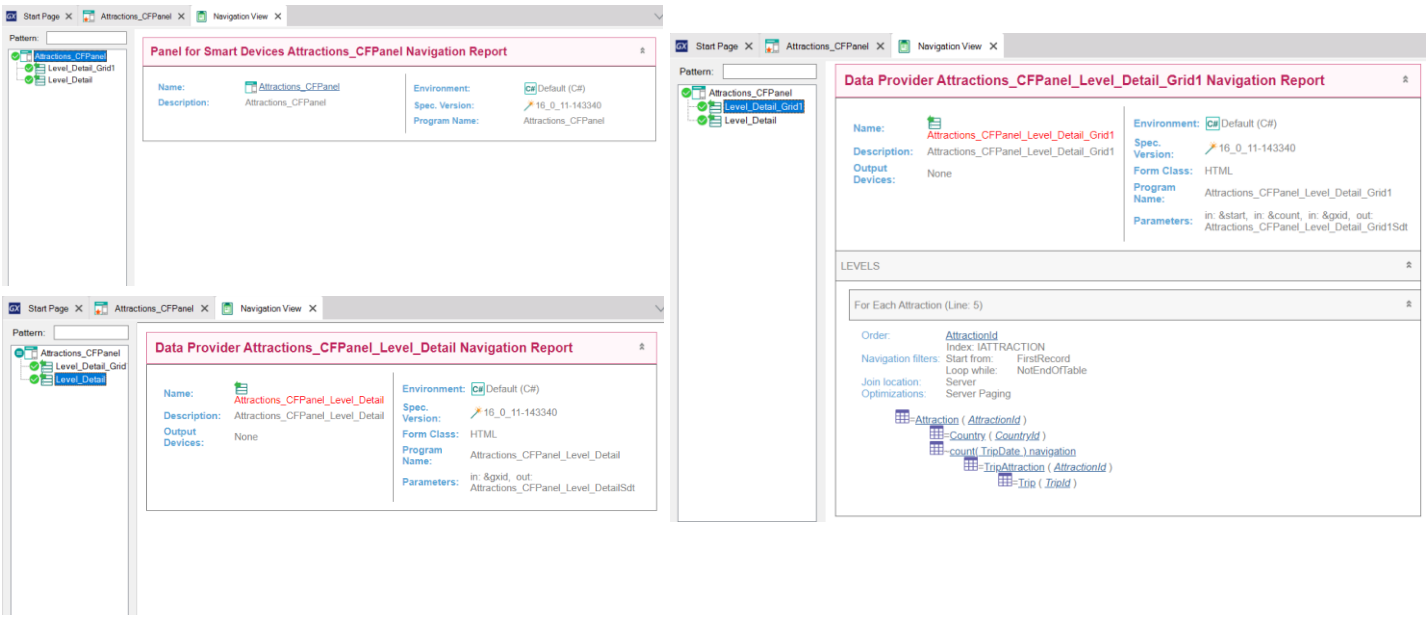
Para obter a quantidade de viagens de uma atração, é preciso percorrer a tabela TripAttraction. Como isto está relacionado à base de dados, temos que programar um evento que se dispare no servidor. Nos objetos panels também contamos com os eventos Start, Refresh e Load como nos webpanels.

Vamos programar os eventos como fizemos com o web panel WWAttractionsFromScratch. No evento Load usamos uma fórmula Count que mediante o atributo TripDate conte os registros das viagens, como vimos antes, embora o atributo TripDate seja da tabela Trip, GeneXus não escolherá a tabela Trip como tabela base da fórmula, mas a tabela TripAttraction.
Como sabemos que o grid percorre a tabela Attraction porque o vimos na lista de navegação, o evento Load será disparado uma vez para cada linha do grid, pelo que a fórmula count contará as viagens de cada atração mostrada. Então, acumulamos em &TotalTrips o total de todas as viagens.
Cabe o mesmo esclarecimento que vimos antes, neste caso usamos o evento Load geral porque temos um único grid, mas poderíamos ter usado o evento Grid1.Load, o que nos permite que no futuro possamos adicionar outro grid ao panel e não mude a programação.

Como queremos que o total de viagens seja inicializado em zero antes de começar a contar as viagens das atrações, vamos adicionar no evento Refresh a inicialização para a variável &TotalTrips, de forma análoga ao que havíamos feito no webpanel.

Vamos executar novamente, então clicamos com o botão direito sobre o panel e selecionamos Run.

Listas de navegação



Se analisamos a lista de navegação, vemos que aparece um novo nível de detalhe diferente do grid, chamado Level_Detail. Aqui é onde se mostra a carga da parte fixa do panel, ou seja, de todos os controles do form que não estão incluídos em um grid.

Ao contrário dos web panels, em um objeto panel a parte fixa é independente do grid e até mesmo, como veremos mais adiante, a parte fixa pode ter uma tabela base diferente da tabela base do grid.

A lista de navegação do nó Level_Detail (que invoca o data provider mencionado no título) aparece vazia, porque não está sendo carregado nenhum campo da base de dados, somente está a variável &TotalTrips, que se atualiza dentro do evento Load.

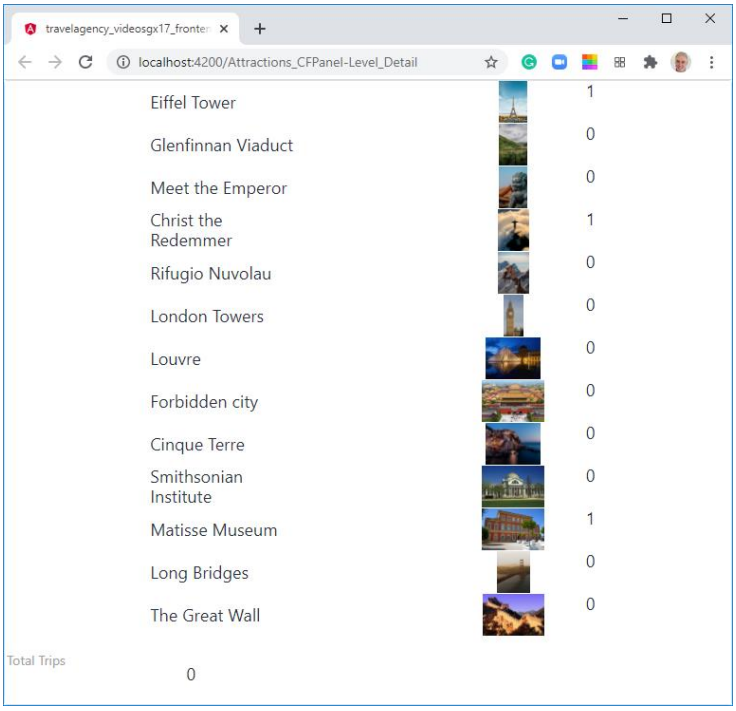
Se formos à lista de navegação do Level_Detail_Grid1, vemos que o data provider correspondente está acessando a tabela Attraction, ordenada por AttractionId que é a ordem padrão e que também aparece a navegação da fórmula Count já que a mesma é disparada no evento Load.

Total global de viagens mal calculado!

Layout * | Rules | **Events *** | Conditions | Variables

Events

```
1  Event Load
2       &Trips = Count(TripDate)
3       &TotalTrips = &TotalTrips + &Trips
4  Endevent
5
6  Event Refresh
7       &TotalTrips = 0
8  Endevent
9
```



Se vamos para a aplicação executando no navegador, vemos que aparece o total de viagens de cada atração corretamente, mas que o total acumulado de viagens sai em 0.
O que fizemos de errado? A variável &TotalTrips está sendo incrementada no evento Load como fizemos no webpanel e temos a segurança de que este evento está sendo disparado porque vemos que algumas atrações têm viagens... Então?

A razão é que os objetos panels não funcionam como os web panels. Nos objetos panel, a parte fixa é carregada de forma independente do grid.
Neste caso, a variável &TotalTrips está na parte fixa do painel, que é o que se carrega primeiro e então ocorre a carga do grid, portanto quando se mostra a variável ainda não pôde carregar o grid, ainda não foi disparado o evento Load e não foi possível acumular o valor de &TotalTrips.

Recordemos que em um objeto panel usado para desenvolver aplicações com foco em customer-facing, para carregar a tela no dispositivo cliente, invoca-se a serviços localizados no servidor que são os que acessam a base de dados. Estes serviços são data providers, que são independentes para a parte fixa e para o grid (ou cada grid) da tela.

Quando a execução do panel é iniciada, é disparado um evento local no cliente que invoca o data provider para carregar a parte fixa e faz com que sejam disparados os eventos Start e Refresh no servidor. Em seguida, é executado um segundo data provider que dispara o

evento Load no servidor N vezes e carrega o grid.

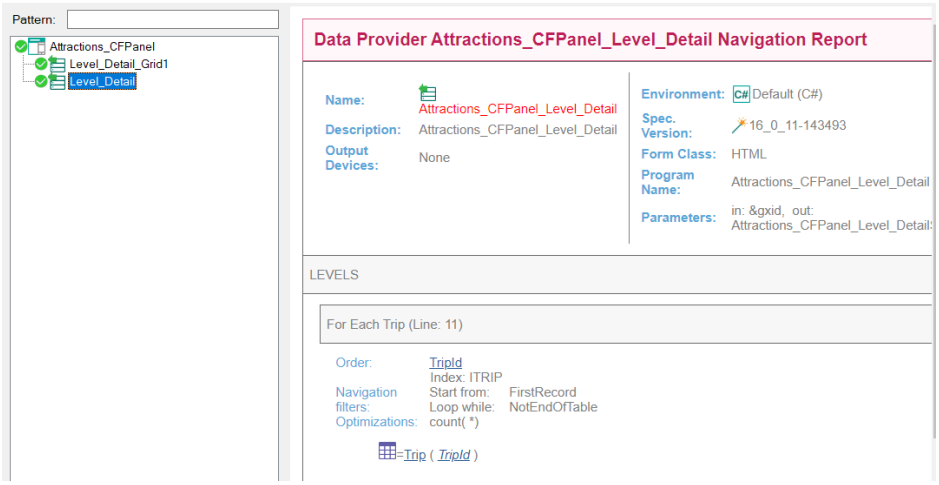
Em nosso exemplo, ao ser disparado o Refresh primeiro, inicializa-se a variável &TotalTrips e carrega-se a parte fixa, logo depois é disparado o evento Load que é onde &TotalTrips é carregado com o valor correto e atualiza-se o grid, mas a parte fixa já foi carregada antes e não volta a ser desenhada.

Devido a esta característica não podemos programar o objeto panel como se fosse um webpanel.

Em outro vídeo entraremos em detalhes no disparo dos eventos e na determinação das tabelas base, mas por enquanto, para que o exemplo funcione, vamos mudar a programação.

Solução correta

```
1  Event Load
2       &Trips = Count(TripDate)
3  Endevent
4
5  Event Refresh
6       &TotalTrips = 0
7       For each Trip.Attraction
8            &TotalTrips += 1
9       Endfor
10 Endevent
```



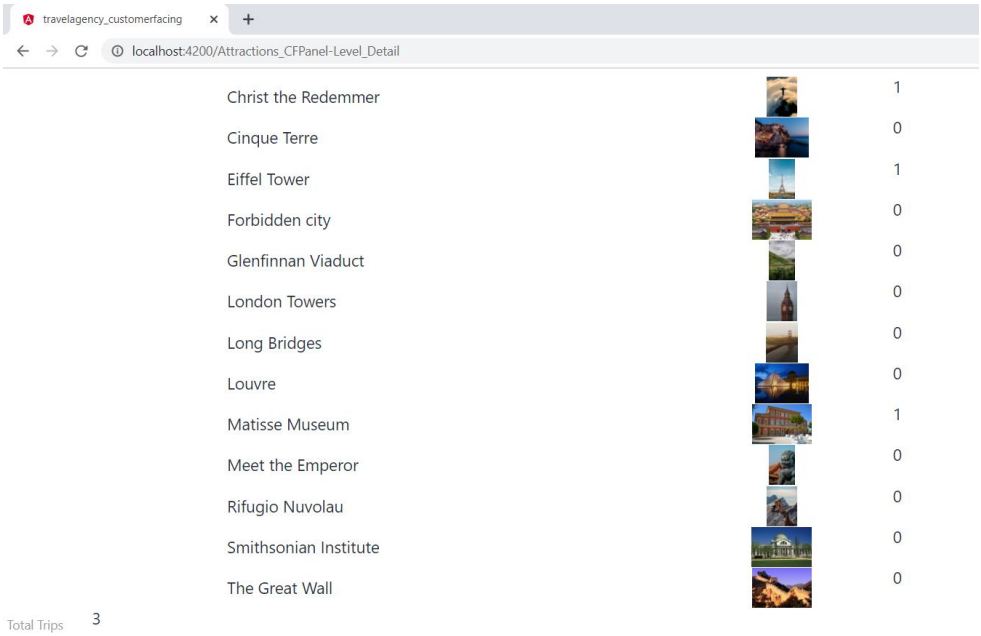
A solução é incluir no evento Refresh um For Each que acesse a tabela TripAttraction e conte o total global de viagens. Não nos esqueçamos de eliminar o cálculo que tínhamos no evento Load.

A razão pela qual incluímos a atualização no evento Refresh, utilizando um For Each, é porque o evento Refresh será disparado quando for carregada a parte fixa, que será antes de ser carregado o grid.
Portanto, ao carregar a parte fixa o valor de &TotalTrips terá o valor correto e logo depois se atualizará a parte do grid.

Executamos...

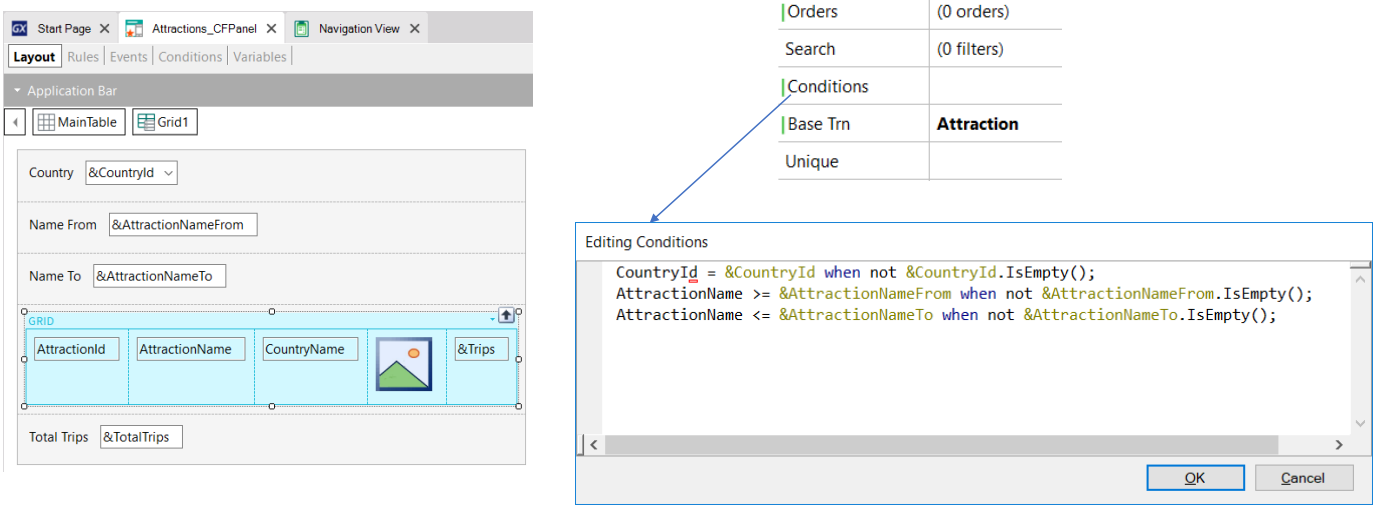
Observamos que agora a lista de navegação correspondente ao nó Level_Detail, inclui o For Each com a navegação na tabela TripAttraction.

Execução com o total acumulado de viagens correto



No navegador, vemos que agora é exibido corretamente o total acumulado de viagens.

Adicionamos filtros por país e nome de atração

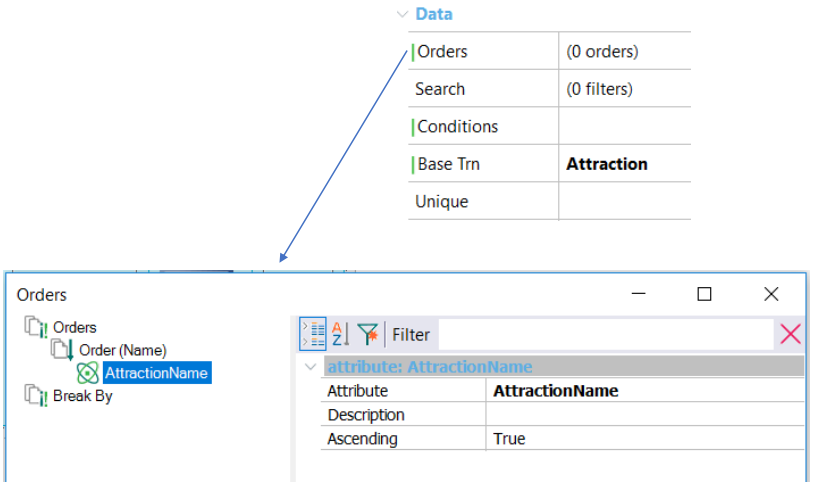


Para completar o panel de trabalhar com atrações, faltaria adicionar os filtros pelo identificador de país e por nomes de atração.

Adicionamos acima do grid uma variável &CountryId como Dynamic Combobox, com a propriedade Item Descriptions que mostre o CountryName e definimos a propriedade Empty Item em True. Adicionamos também as variáveis &AttactionNameFrom e &AttractionNameTo para filtrar as atrações por nome.

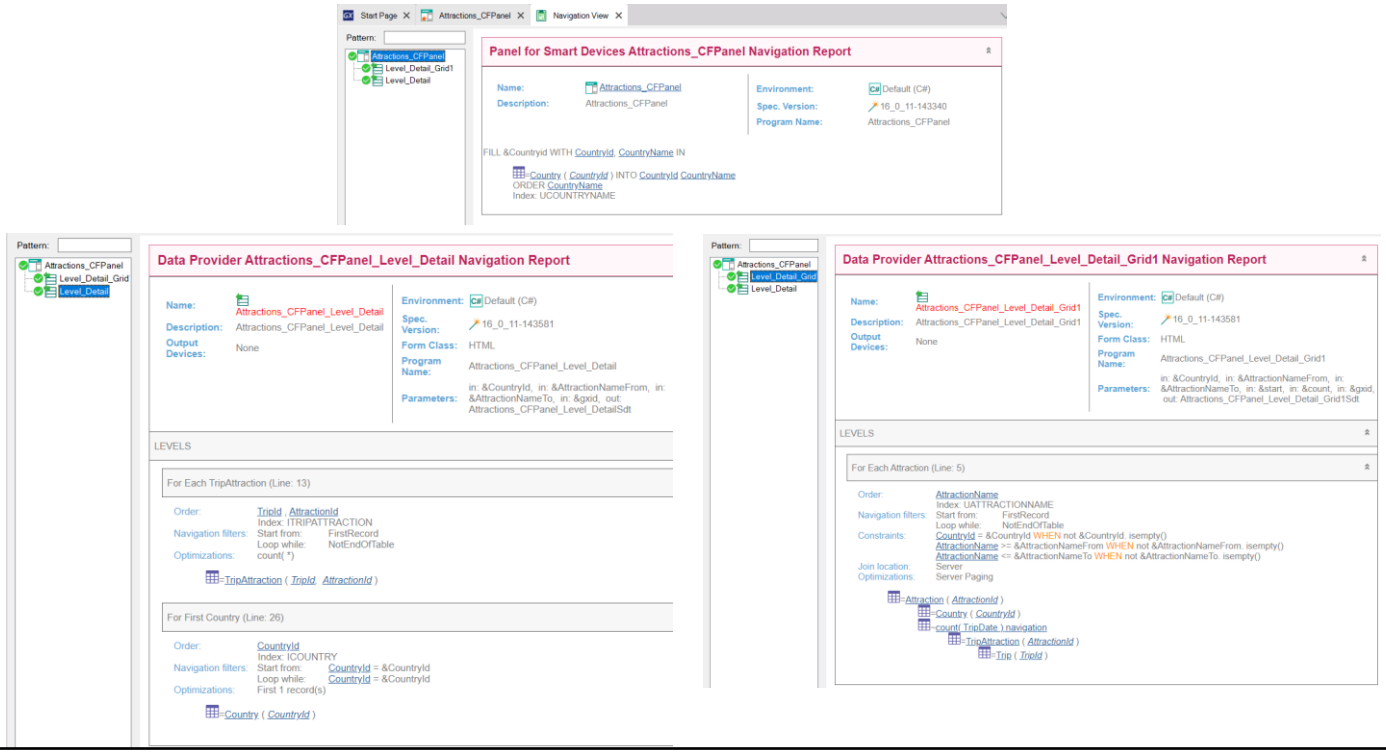
Em um webpanel adicionaríamos o filtro nas conditions do grid, aqui também podemos fazer o mesmo. Definamos uma condition para filtrar por país e outras para filtrar desde e até o nome de atração.

Ordenamos a lista de atrações por nome



Também dispomos de uma propriedade Order no grid, onde vamos indicar que o grid saia ordenado por nome de atração, então clicamos com o botão direito sobre Orders, lhe damos um nome, por exemplo, Name e então clicamos outra vez com o botão direito para inserir o atributo AttractionName.

Podemos definir uma ordem por vários atributos, assim como também criar outras ordens por diferentes critérios. A parte onde diz Break By, nos permite agrupar os registros do grid, por exemplo, se quiséssemos que as atrações saiam agrupadas por nome de país.

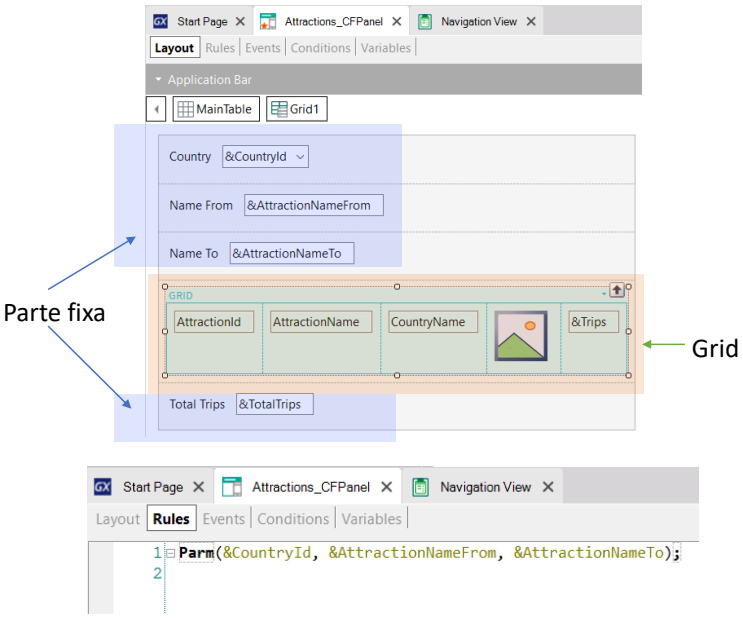


Se clicamos com o botão direito sobre o panel e selecionamos View Navigation, na lista de navegação do panel mesmo, vemos que se acessa a tabela Country para preencher o Dynamic Combobox (onde diz FILL &CountryId WITH CountryId, CountryName IN).

A lista de navegação do nó Level_Detail nos mostra o relatório do Data Provider (criado automaticamente por GeneXus mas que não é visto na KB) que é invocado no servidor para recuperar os dados da base de dados, necessários para carregar a parte fixa do painel. Este data provider dispara no servidor o evento Start e o Refresh.
Na lista vemos o For each que programamos no evento Refresh, que acessa a tabela TripAttraction para contar o total de viagens e vemos também o acesso à tabela Country filtrada pelo CountryId selecionado no Dynamic combo.

Se vemos a lista de navegação correspondente ao data provider que carrega o grid, vemos que agora as atrações serão percorridas ordenadas por AttractionName e que nas constraints aparecem os filtros que definimos. Este data provider executa internamente o evento Load uma vez para cada linha do grid a ser carregada como nos webpanels, se o grid tem tabela base e devolve a informação ao panel para ser mostrada.

Necessidade de atualizar o conteúdo com os filtros




```
1  Event Load
2      &Trips = Count(TripDate)
3  Endevent
4
5  Event Refresh
6      &TotalTrips = 0
7      For each Trip.Attraction
8          &TotalTrips += 1
9      Endfor
10 Endevent
11
12 Event &CountryId.ControlValueChanged
13     Grid1.Refresh()
14 Endevent
15
16 Event &AttractionNameFrom.ControlValueChanged
17     Grid1.Refresh()
18 Endevent
19
20
21 Event &AttractionNameTo.ControlValueChanged
22     Grid1.Refresh()
23 Endevent
```

Uma das coisas que mencionamos foi que a parte fixa do panel é carregada de forma independente da carga do grid, invocando data providers diferentes, que estão publicados no servidor como serviços e acessam a base de dados para recuperar as informações de cada parte.

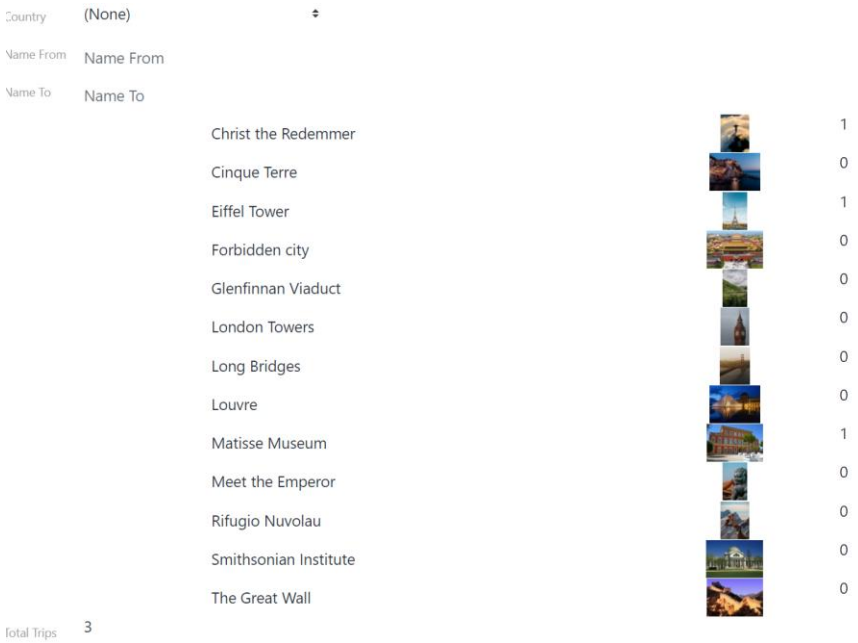
Quando alteramos o valor de um filtro, é necessário que seja atualizada a informação do grid. Para isto, é necessário adicionar o método Refresh do grid, que disparará os eventos Refresh e Load do servidor, o que fará com que se carregue novamente o grid, aplicando as conditions programadas e mostre os resultados filtrados como esperamos.

Como o método Refresh do grid devemos invocá-lo depois que mudamos o valor da variável do filtro, usamos o evento ControlValueChanged de cada variável, para invocar o método. Desta forma, depois de mudar um valor, ao sair do campo será disparado o evento correspondente que terminará atualizando o conteúdo do grid.

No entanto, há outra coisa que devemos levar em conta e é que nesta arquitetura, como o objetivo é que a página seja carregada a menor quantidade de vezes possível, prioriza-se o cache de dados, ou seja, que trata-se sempre de recuperar a informação previamente armazenada. Para que o servidor entenda que queremos trazer novos dados, é preciso fazer com que se altere a URL enviada ao servidor, de forma que este interprete que é uma página nova e obtenha a informação correspondente para enviá-la ao cliente.
Para isso, adicionamos uma regra Parm, que contenha os valores das variáveis que usamos nos filtros, de forma que, se muda o valor de uma variável, se atualize a página com os novos dados.

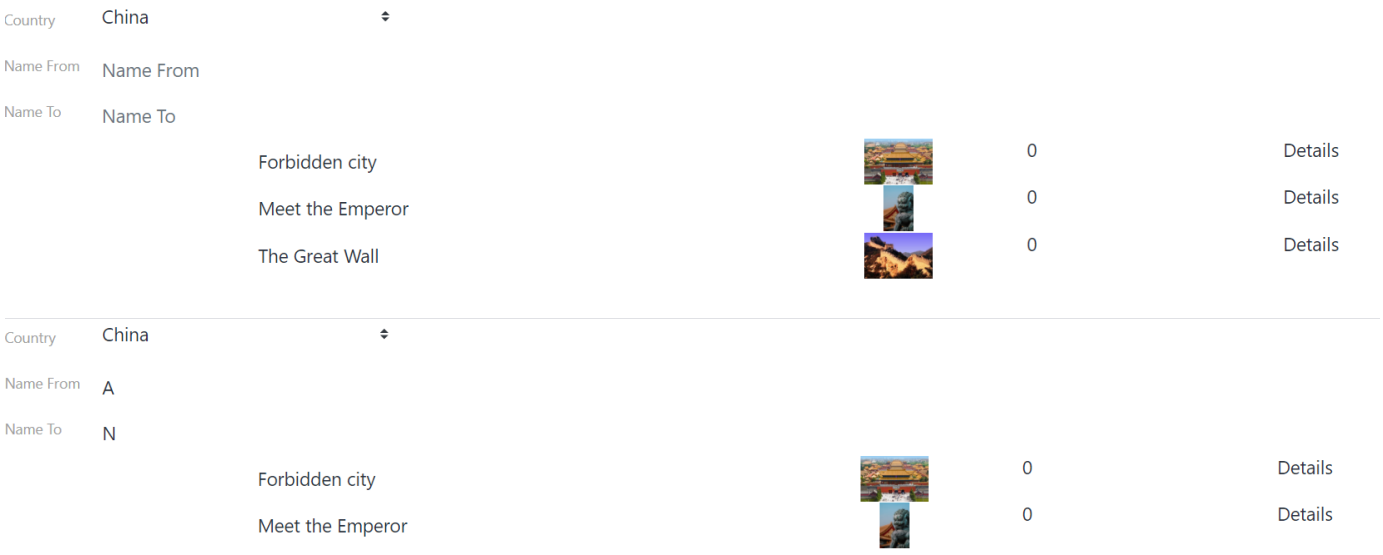
Agora sim, executamos para testar tudo isto que vimos.

Execução com os novos filtros e ordem



Vemos que na parte superior agora aparecem os filtros que adicionamos e as atrações estão ordenadas por nome como esperávamos.

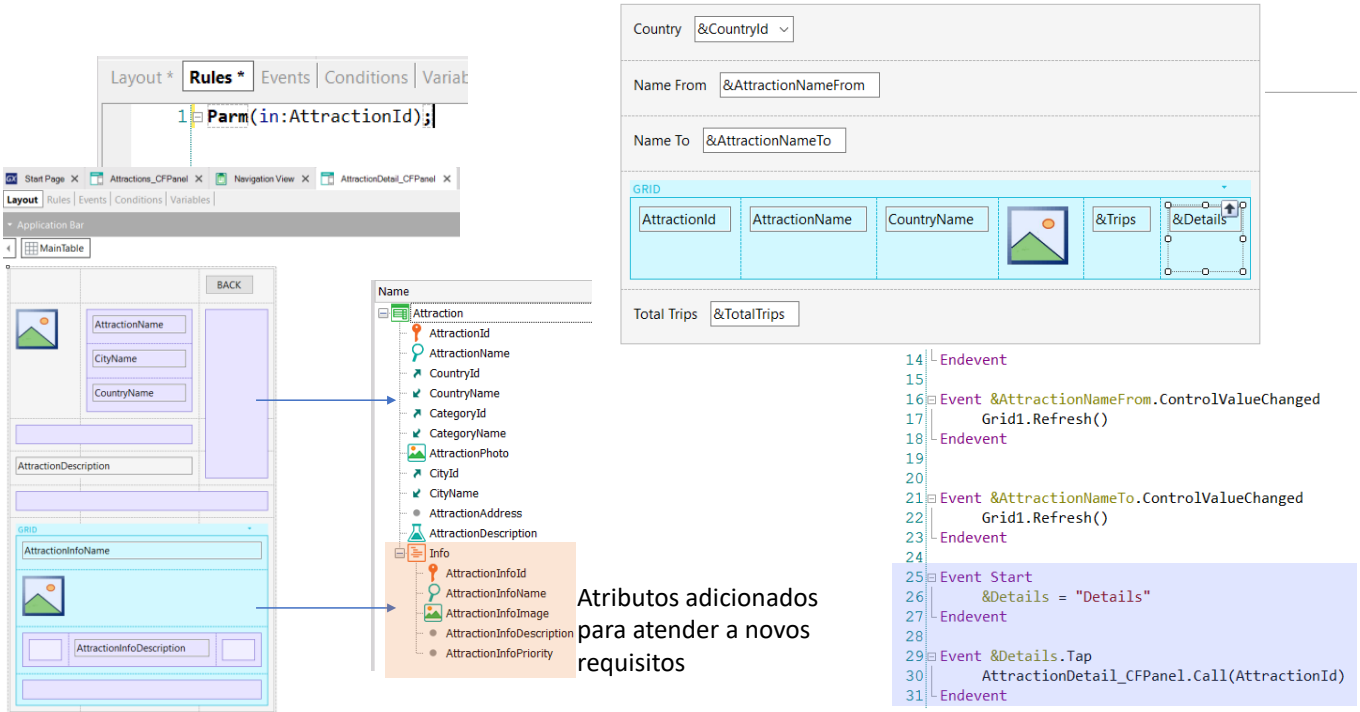
Execução com os novos filtros e ordem (cont.)



Vamos filtrar pelo país China e vemos que nos mostra as atrações da China.

Agora, escolhemos ver as atrações que começam com a letra A até a letra N e somente vemos as atrações da China com o nome nesse intervalo.

Implementamos o detalhe de uma atração



Atributos adicionados para atender a novos requisitos

Agora, completemos o requisito pedido pela agência, que ao clicar em uma das atrações listadas, seja mostrado o detalhe da mesma. Para isso utilizaremos outro objeto panel de nome AttractionDetail_CFPanel. Para economizar tempo, eu já o criei.

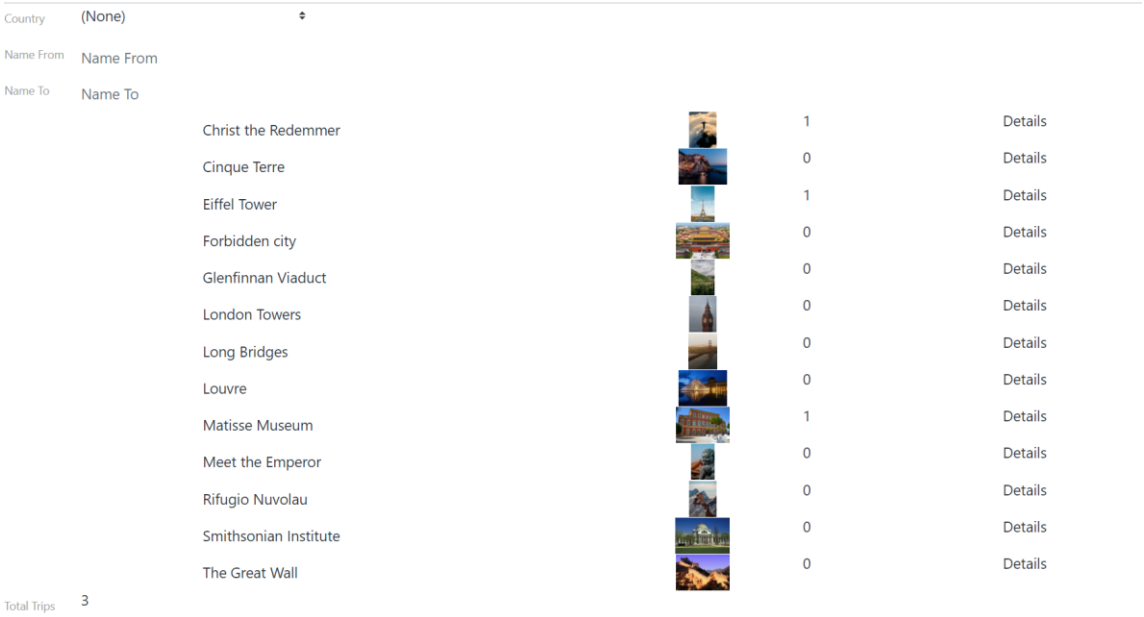
Vemos que nas regras definimos uma regra Parm, com um parâmetro de entrada, o atributo AttractionId.
Recordemos que isto permitirá que seja mostrada somente a informação da atração que passemos por parâmetro, que foi a que selecionamos no panel da lista de atrações.

No form colocamos o atributo AttractionPhoto e os atributos AttractionName, CityName e CountryName. Mais abaixo inserimos o atributo AttractionDescription. Depois inserimos um grid e selecionamos os atributos AttractionInfoName, AttractionInfoImage e AttractionInfoDescription para ver a informação da atração. Como podemos ver, inserimos alguns controles Table para alinhar melhor.
Também adicionamos acima à direita, um botão BACK que invocará um Return que nos permitirá retornar à lista de atrações.

Agora vamos ao panel Attractions_CFPanel e adicionamos uma variável &Detail do tipo Character ao grid. No evento Start atribuímos-lhe o texto "Details".
Depois, inserimos um evento para a variável &Detail do Grid, selecionamos Tap e escrevemos a invocação para AttractionDetail_CFPanel, passando-lhe como parâmetro AttractionId.

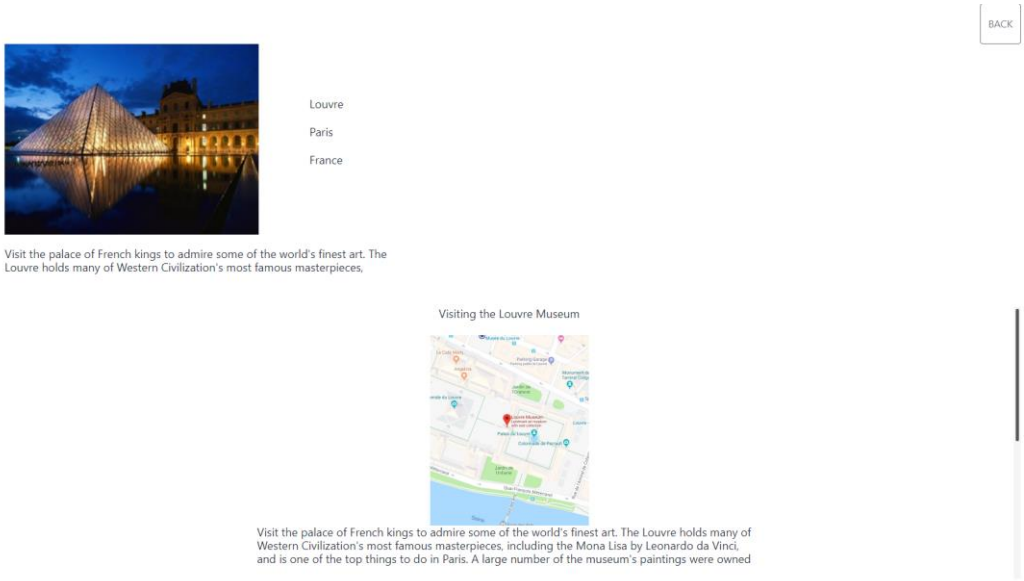
Testemos isto em execução.

Execução com o detalhe de uma atração



Vamos ver as informações do museu do Louvre, por isso na linha correspondente clicamos em Details.

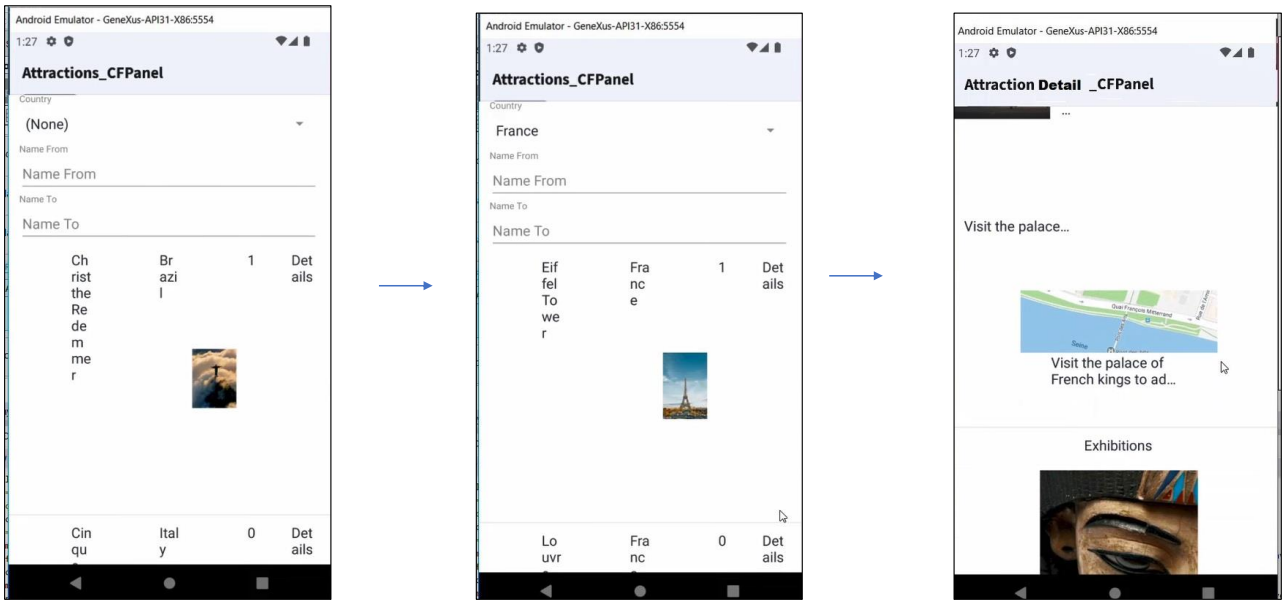
Execução com o detalhe de uma atração (cont.)



O panel é aberto com o detalhe da atração, onde podemos observar um mapa e as exibições disponíveis.

Gerando a aplicação em Android nativo

Pré-requisitos para Android https://wiki.genexus.com/commwiki/servlet/wiki?14449



Pois bem... No início deste vídeo dissemos que se quiséssemos, poderíamos usar os objetos panel que construímos para gerar as telas de nossa aplicação móvel. Vamos testar isto.

Para isso, no KB Explorer, clicamos no nó Front end e colocamos a propriedade Generate Android como True. Agora executamos nosso objeto main Attractions_CFPanel.

Vemos que se abre um emulador de Android exibindo o panel Attractions_CFPanel, com a lista de atrações turísticas.
Obviamente não parece muito bom, pois quando implementamos este panel estávamos pensando em um sistema web, que rodaria na tela de um notebook ou um computador desktop e deveríamos definir um desenho de acordo com o tamanho da tela de um telefone.

Mas além do desenho, vamos verificar se funciona corretamente. No filtro escolhemos France e vemos que são mostradas apenas as atrações da França. Agora clicamos em Details do Louvre e vemos que se abre a tela com os detalhes do museu, onde vemos o mapa e as exposições como esperávamos.

Neste vídeo começamos a nos familiarizar com os objetos panel, gerando a aplicação no framework Angular e verificamos que com os mesmos objetos criados pudemos gerar também a aplicação em linguagem nativa Android.

A seguir, conheceremos mais sobre o manejo de eventos no nível do cliente e do servidor.

GeneXus™

training.genexus.com
wiki.genexus.com