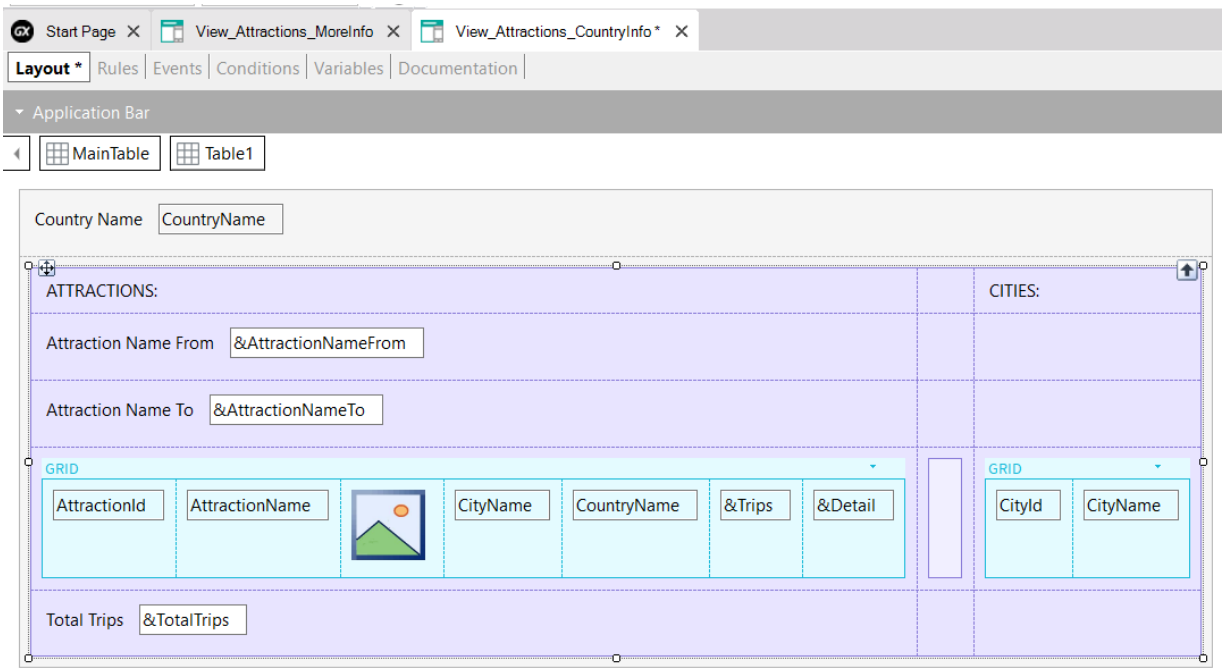
Panel com múltiplos grids

GeneXus™

Vimos como usar um grid em um objeto panel. Veremos agora as considerações que devemos ter se adicionamos ao panel mais de um grid.
Isto nos fornecerá os conhecimentos necessários para continuar avançando no desenvolvimento de nossa aplicação para a agência de viagens.

Objeto panel com mais de um grid



Vamos construir um objeto panel que mostre as atrações e as cidades de um país recebido por parâmetro.

Para isso, fazemos um Save As do panel View_Attractions_MoreInfo e o chamamos de View_Attractions_CountryInfo. Em seguida, retiramos que seja objeto main, pois a ideia é que seja chamado a partir do grid de atrações ao clicar sobre o nome de um país.

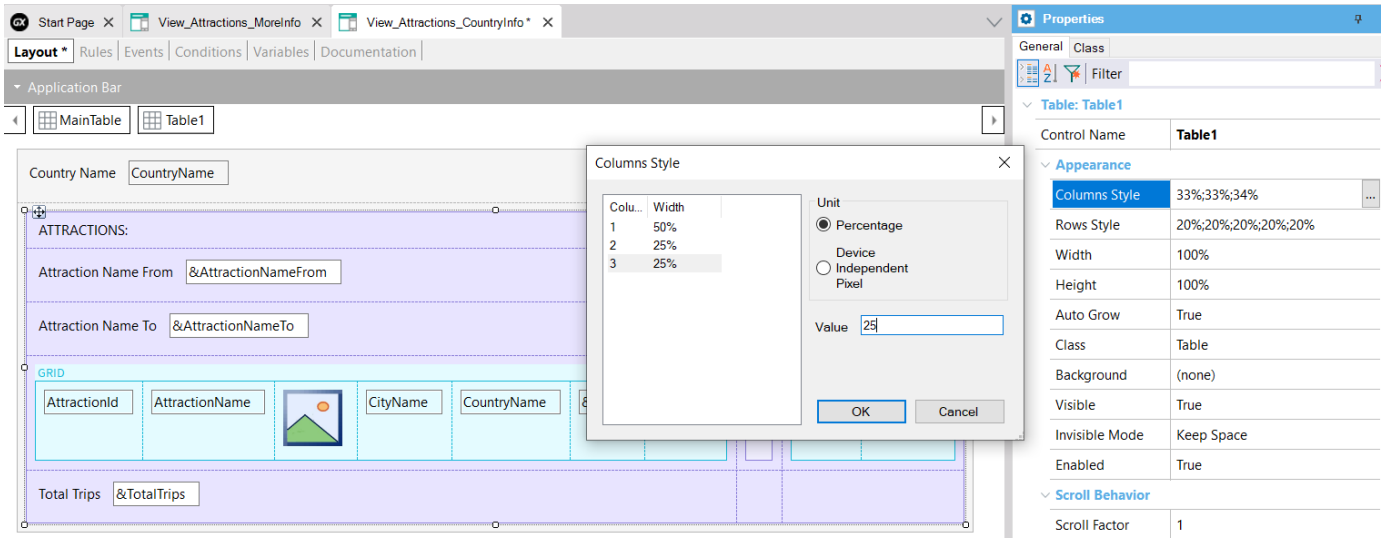
No form, vamos eliminar os títulos que tínhamos, pois mostraremos títulos por seção e no grid vamos adicionar uma variável &Detail, do tipo Character. Esta variável nos servirá posteriormente para invocar um panel que mostre o detalhe de cada atração.
Nos eventos, retiramos as linhas que atribuíam o texto aos títulos antigos e atribuímos à variável &Detail, a legenda “Details”.

No form adicionamos o atributo CountryName e eliminamos a variável do tipo Dynamic combo &CountryId, pois não vamos escolher um país, mas o recebemos por parâmetro. Removemos a variável das conditions. Eliminamos o evento ControValueChanged associado à variável que removemos, a cláusula where que a utilizava e na regra Parm alteramos a variável &CountryId para o atributo CountryId

Para agrupar todos os dados das atrações e depois os dados das cidades, inserimos um controle Table a partir da barra de ferramentas e colocamos as variáveis AttractionNameFrom, AttractionNameTo, o grid e a variável TotalTrips dentro da tabela. Agora à sua direita inserimos outra tabela como separador e a seguir inserimos um grid para mostrar as cidades do país, com os atributos CityId e CityName.

Adicionamos títulos às seções, colocando um textblock ATTRACTIONS: acima da seção de atrações e CITIES: acima da seção de cidades. Mudamos a classe para TextBlockTitle.

Alterando o estilo das linhas e colunas de uma tabela



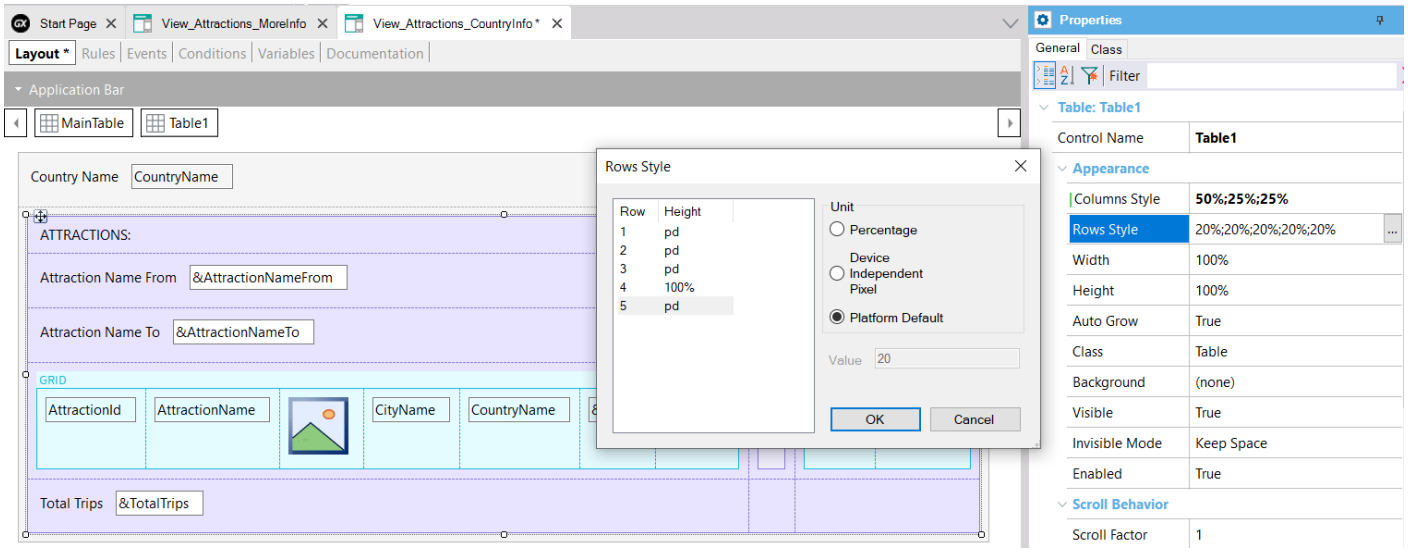
Para definir como queremos que o conteúdo de uma tabela seja visto, devemos alterar o tamanho das linhas ou das colunas da tabela. Por exemplo, queremos que os grids tenham espaço suficiente na linha correspondente para poder ser exibido corretamente e atribuir mais espaço à coluna da tabela que mostra o conteúdo das atrações, porque temos mais coisas para mostrar do que para as cidades.

Para isso contamos com as propriedades Columns Style e Rows Style da tabela. Primeiro vamos mudar as colunas que sabemos que são 3, a coluna que contém os dados das atrações, a coluna com a tabela separadora e a coluna que contém os dados das cidades. Se selecionamos a Table1 e clicamos na propriedade Columns Style, vemos que por padrão cada coluna ocupa 33% do espaço disponível.

Vemos que os valores possíveis para serem atribuídos à largura das colunas estão em porcentagem do espaço total da tabela ou em Device Independent Pixels (DIPs). Esta medida permite atribuir unidades que são uma abstração de um pixel, que não dependem da plataforma e que então serão convertidas em pixels reais no momento em que aplicação for executada. A quantidade de pixels que cada DIP ocupará dependerá das dimensões da tela. Isto nos permite dimensionar para diferentes tamanhos de tela usando tamanhos uniformes.

Vamos atribuir à primeira coluna 50%, à segunda 25% e à terceira outros 25% do espaço disponível. Este espaço é o que fica depois de ter atribuído as colunas com dips, ou seja, as porcentagens são relativas ao valor que resulta da subtração da largura total, os valores fixos (em dips).

Alterando o estilo das linhas e colunas de uma tabela

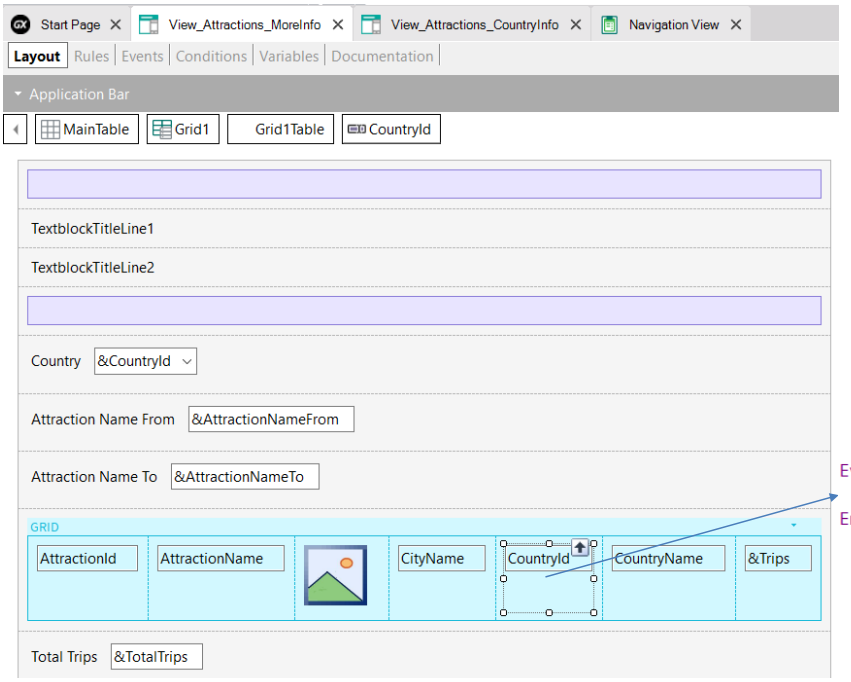


Agora mudamos a altura das linhas. Clicamos na propriedade Rows Style e vemos que aqui podemos atribuir os valores em porcentagem, em DIPS e em Platform Default.

Este último valor difere de plataforma para plataforma e para uma mesma plataforma também depende do conteúdo da célula.

Vamos atribuir esse valor a todas as linhas, exceto para a linha 4, onde estão os grids, que atribuímos para ocupar os 100% que fique disponível.

Invocação do panel de detalhe




```
Event CountryName.Tap
    View_Attractions_CountryInfo(CountryId,"","")
Endevent
```

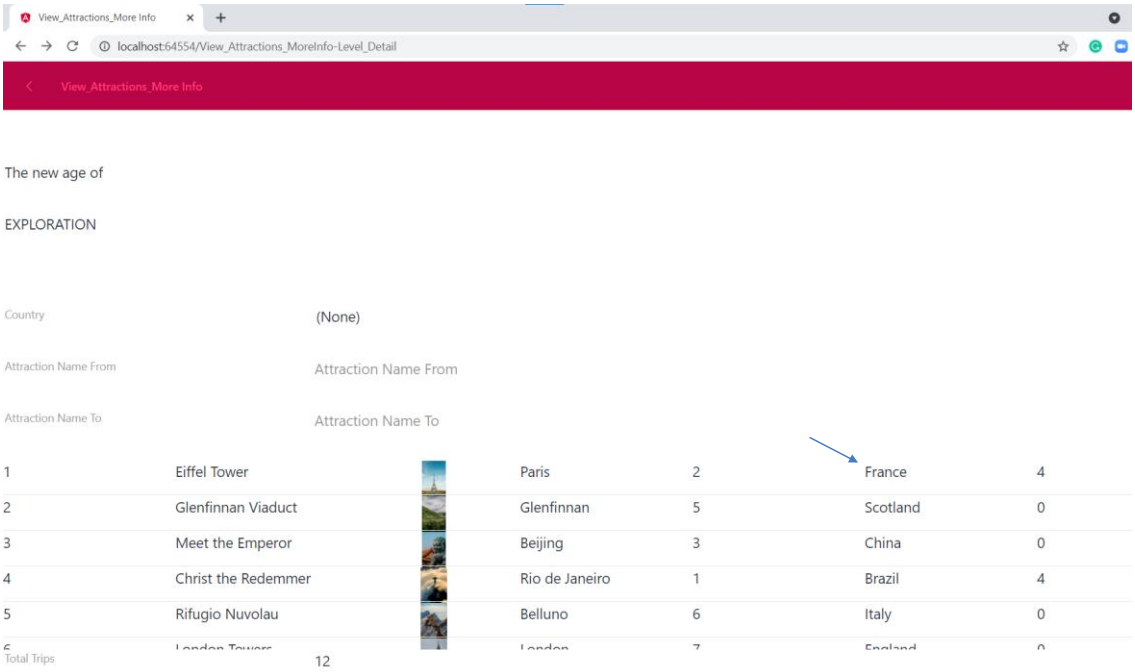
Se clicamos com o botão direito para ver a navegação do objeto que construímos, vemos que na janela de Output mostra um erro e se vemos o erro na lista de navegação, nos diz que o evento Load não pode ser programado se tivermos múltiplos grids.
Se formos aos eventos, vemos que ainda temos programado o evento Load como o tínhamos no objeto original. Aqui, não percebemos que teria sido melhor utilizar o evento Load do grid e não o genérico para evitar essa situação.

Portanto, adicionamos Grid1 ao Load. Se fizermos View Navigation, vemos que o problema foi resolvido.

Antes de executar, vamos ao panel View_Attractions_MoreInfo e adicionamos ao grid de atrações o atributo CountryId que precisamos para passar ao panel de informações de um país, quando clicarmos sobre o nome do país no grid.
Em seguida, adicionamos ao atributo CountryName um evento Tap, onde escrevemos a invocação ao panel View_Attractions_CountryInfo, passando o CountryId como parâmetro.

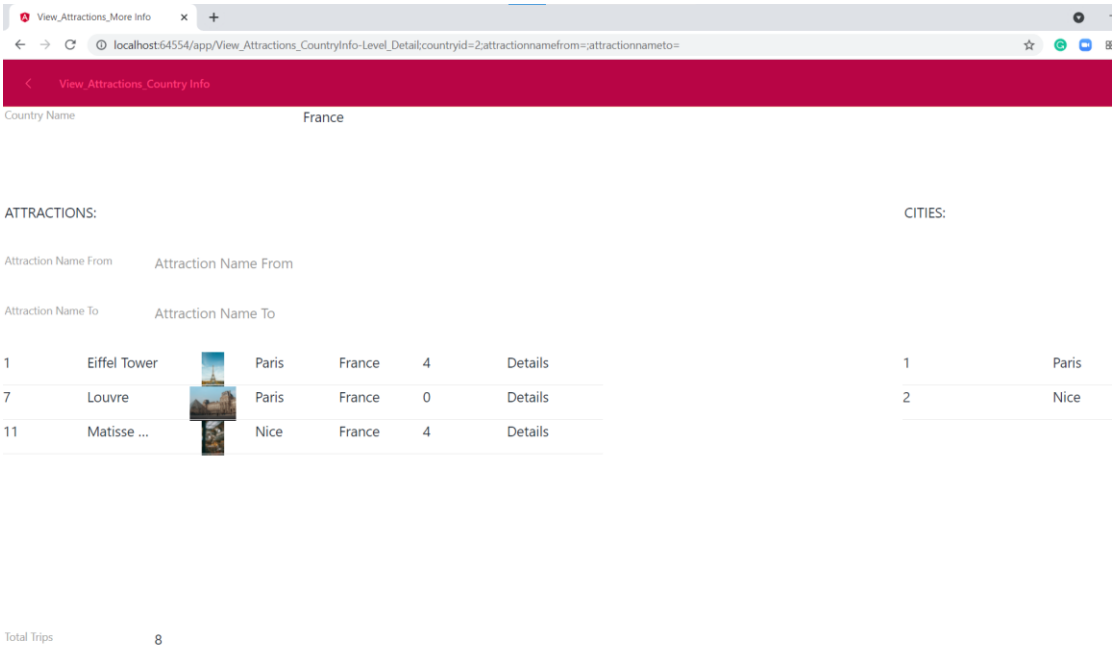
Vamos executar para ver tudo isto.

Exemplo em execução



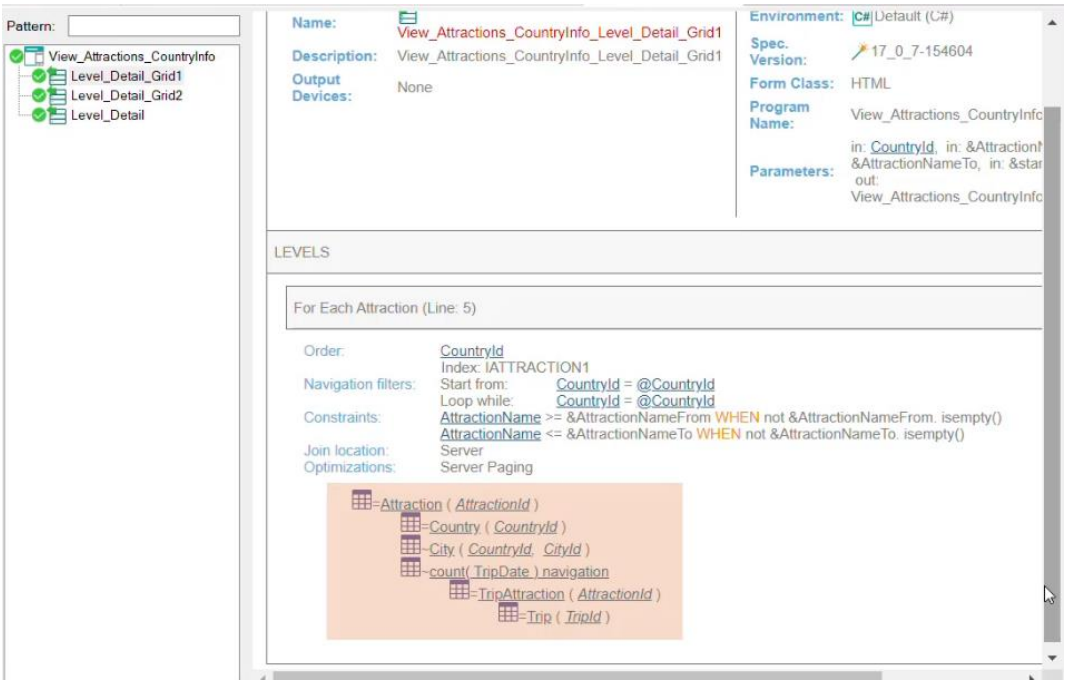
Se clicamos sobre France…

Exemplo em execução



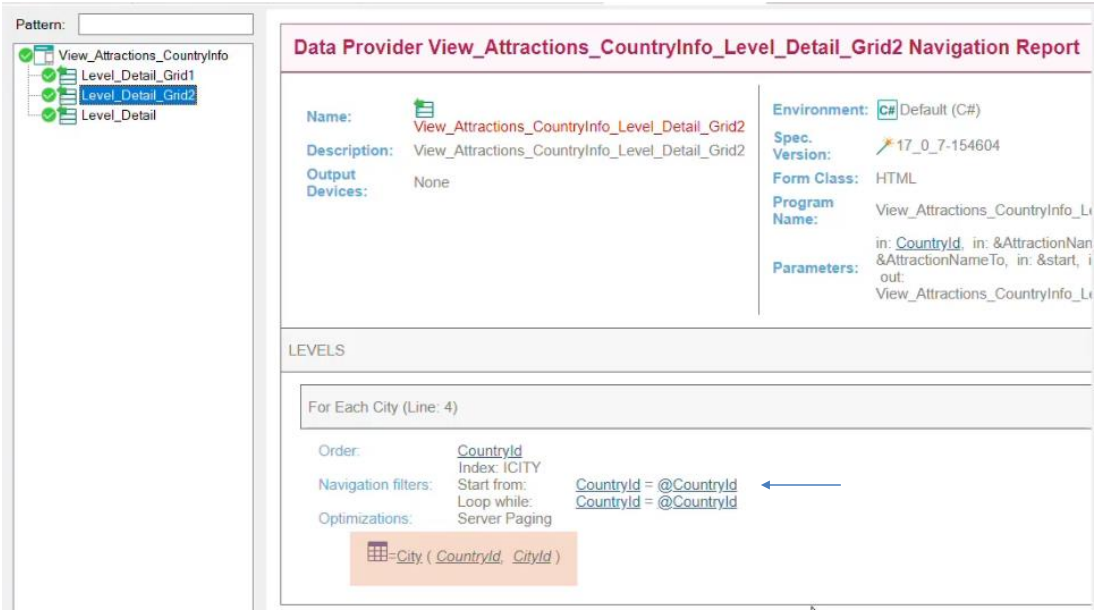
Se abre a informação do país França, mostrando as atrações com o total de viagens de cada atração e o total de viagens para França e, à direita, as cidades desse país.

Lista de navegação do novo panel



Se vamos para a lista de navegação, vemos que agora há uma entrada Level_Detail para o Grid1 e outra para o Grid2.
Para o Grid1 vemos o acesso à tabela Attraction e a navegação da fórmula que vimos quando implementamos o painel View_Attractions_MoreInfo.

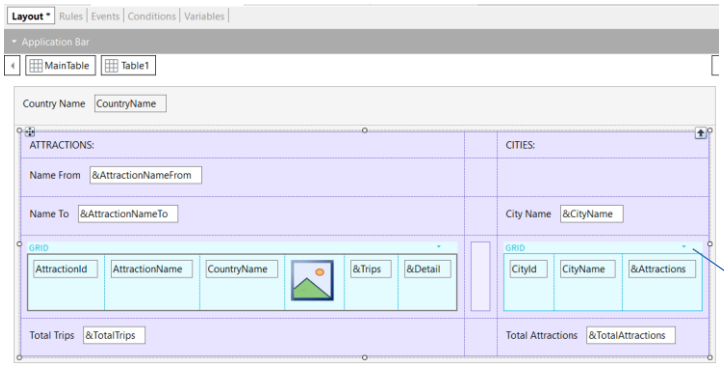
Lista de navegação do novo panel

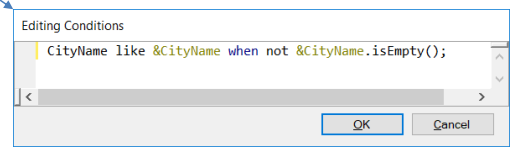


Se vamos ao nó correspondente ao Grid2, vemos que o grid está acessando a tabela City para mostrar as cidades e que está filtrando por CountryId, pelo atributo recebido na regra Parm.

Adicionamos filtro por cidade e totais por cidade e país

```
Event Start
     &Detail = "Details"
Endevent

Event Grid1.Load
     &Trips = Count(TripDate)
Endevent

Event Grid2.Load
     &Attractions = Count(AttractionName)
Endevent

Event Grid1.Refresh
     &TotalTrips = 0
     For each Trip.Attraction
          Where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty()
          Where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty()
          &TotalTrips += 1
     Endfor
Endevent

Event Grid2.Refresh
     &TotalAttractions = 0
     For each Attraction
          &TotalAttractions += 1
     Endfor
Endevent
```





De forma similar aos totais que mostramos para as atrações e o filtro por nome, vamos mostrar a quantidade de atrações que cada cidade possui, o total de atrações do país e vamos adicionar um filtro por nome de cidade.

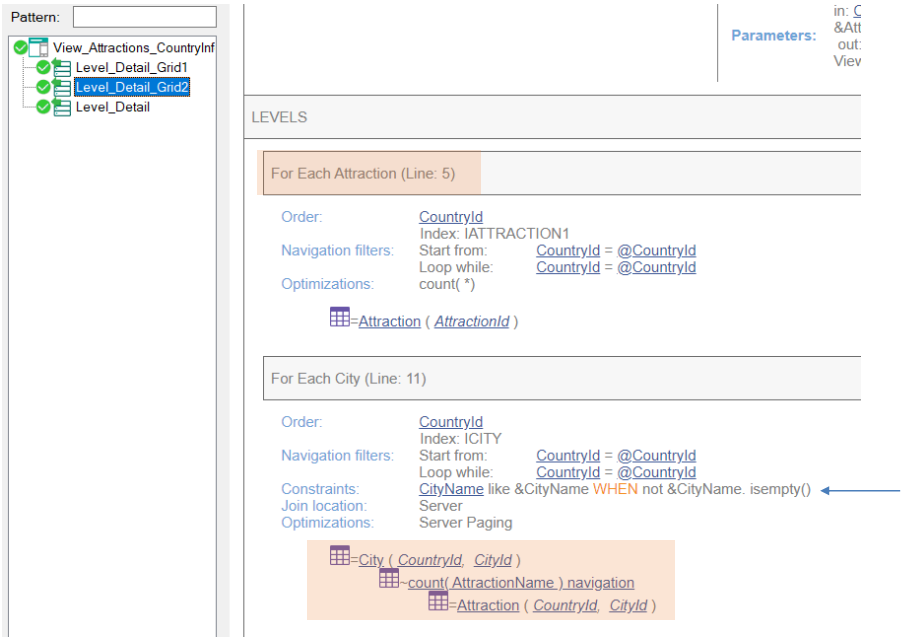
Na guia Variables, criamos uma variável &Attractions, outra &TotalAttractions e outra &CityName.
Agora adicionamos a &Attractions ao grid ajustando sua propriedade Label Position como None. Em seguida, adicionamos &TotalAttractions abaixo do grid e a variável de filtro &CityName acima do grid de cidades. Vamos adicionar ao grid a condição necessária para o filtro.

Antes disso, aproveitamos para atribuir a propriedade Base Transaction em City. Agora sim, clicamos em Conditions e escrevemos a condição de filtro utilizando o operador like, pois não filtraremos por faixa de nome, mas por um nome semelhante.

Em seguida, adicionamos o evento Grid2.Load onde carregamos a variável &attractions com uma fórmula Count com o atributo AttractionName. Como a tabela base do grid é City, a fórmula contará apenas as atrações da cidade correspondente a cada linha.

Para calcular o total de atrações, por motivos idênticos que fizemos quando calculamos o total de viagens, escrevemos o evento Grid2.Refresh, no qual programamos um For Each sobre a tabela de Atrações para contar as atrações, que serão filtradas pelo atributo do identificador do país recebido por parâmetro.

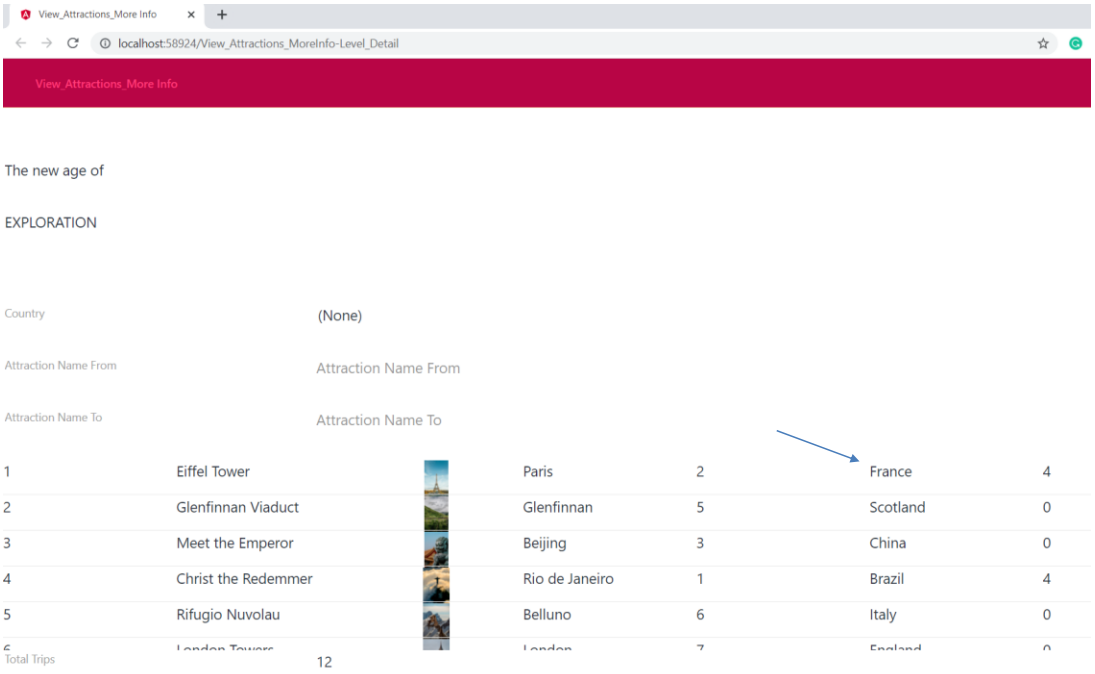
Lista de navegação do panel com as novas alterações



Se vemos agora a lista de navegação do panel, no nó Level_Detail_Grid2 vemos a navegação do For Each para a tabela Attraction e mais abaixo a navegação da fórmula Count sobre a tabela Attraction.

E aqui vemos o filtro que adicionamos por CityName.

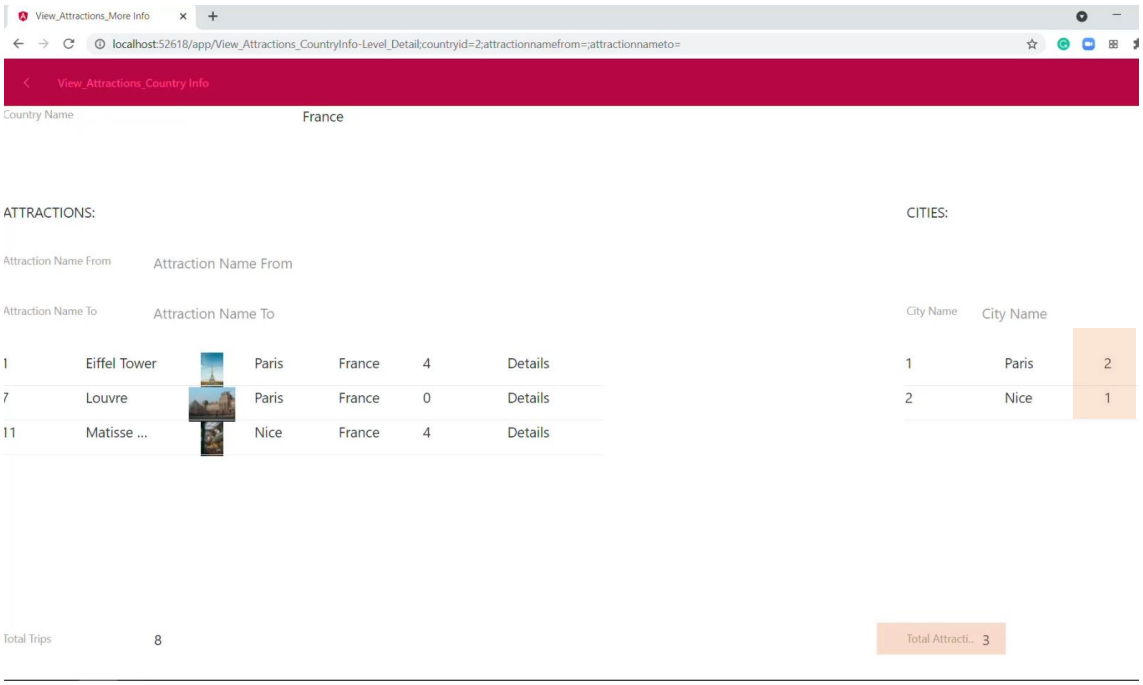
Visualizando os totais por cidade e país



Executemos nosso objeto main, View_Attractions_MoreInfo, para ver o que fizemos.

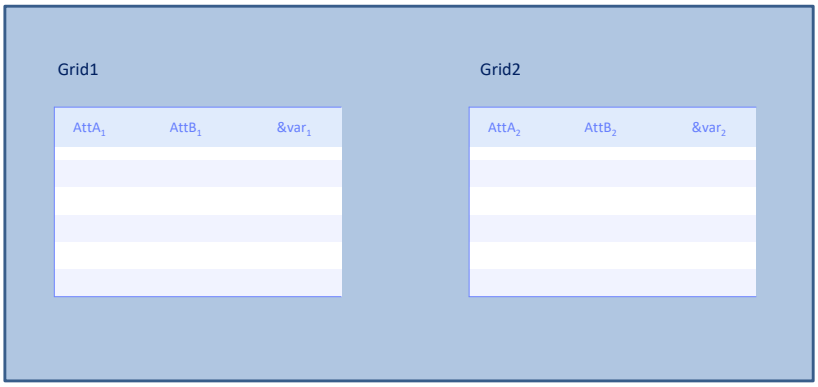
Clicamos em França novamente…

Visualizando os totais por cidade e país



e agora podemos ver o total de atrações de cada cidade da França e o total de atrações do país França.

Event execution order



Start

Refresh

Grid1.Refresh

Grid1.Load

Grid2.Refresh

Grid2.Load

Já vimos que tendo mais de um grid no panel, devemos usar o evento Refresh e Load de cada grid.

Mas, tendo adicionado estes eventos, surge a questão de qual seria a ordem de disparo destes eventos em relação aos eventos próprios do objeto panel.

Ao executar o objeto panel pela primeira vez, a ordem de disparo de execução dos eventos será:

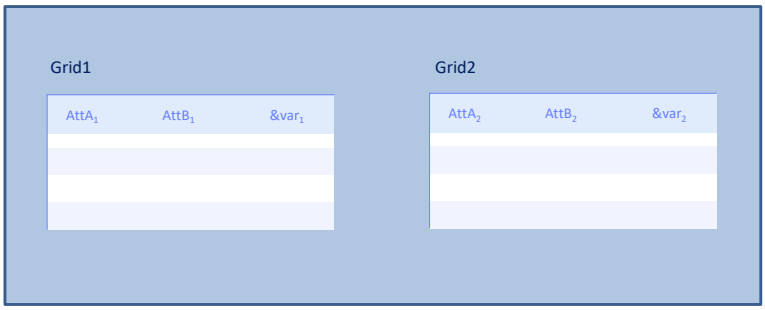
Primeiro o evento Start (apenas uma vez).

Em seguida, o evento Refresh genérico, ou seja, o evento Refresh próprio do panel.

Em seguida, o Refresh do primeiro grid e depois se este possui tabela base, será executado o evento Load do grid tantas vezes quanto registros forem recuperados da base de dados, filtrando os registros que correspondam. Se não houver tabela base, será executado o evento Load do grid apenas uma vez e se for um grid baseado em um SDT, não será executado o evento Load.

E então o mesmo com os eventos Refresh e Load do segundo grid.

What do you want to refresh?



Start

Refresh

Grid1.Refresh

Grid1.Load

Grid2.Refresh

Grid2.Load

Event 'User-event'
...
Form.Refresh
endevent

Event 'User-event'
...
Refresh
endevent

Event 'User-event'
...
Grid1.Refresh()
endevent

Havendo mais de um grid, o comando Refresh também deve ser especializado para indicar qual grid deseja atualizar.

O comando Refresh genérico (aquele que vimos quando o usamos no panel View_Attractions_MoreInfo) faz com que sejam executados o Refresh genérico e o Refresh e Load de cada grid (ou seja, tudo exceto o Start).

E agora temos também o método Refresh de um grid, que irá atualizar apenas esse grid, ou seja, que sejam executados o Refresh do grid e o Load do grid (n vezes, uma vez ou nenhuma), dependendo se o grid tem tabela base, não tem ou é um grid de uma variável SDT coleção, respectivamente.

Neste vídeo vimos como podemos trabalhar com múltiplos grids em um objeto Panel, neste caso com grids paralelos e as considerações que devemos ter na invocação dos eventos de cada grid.

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