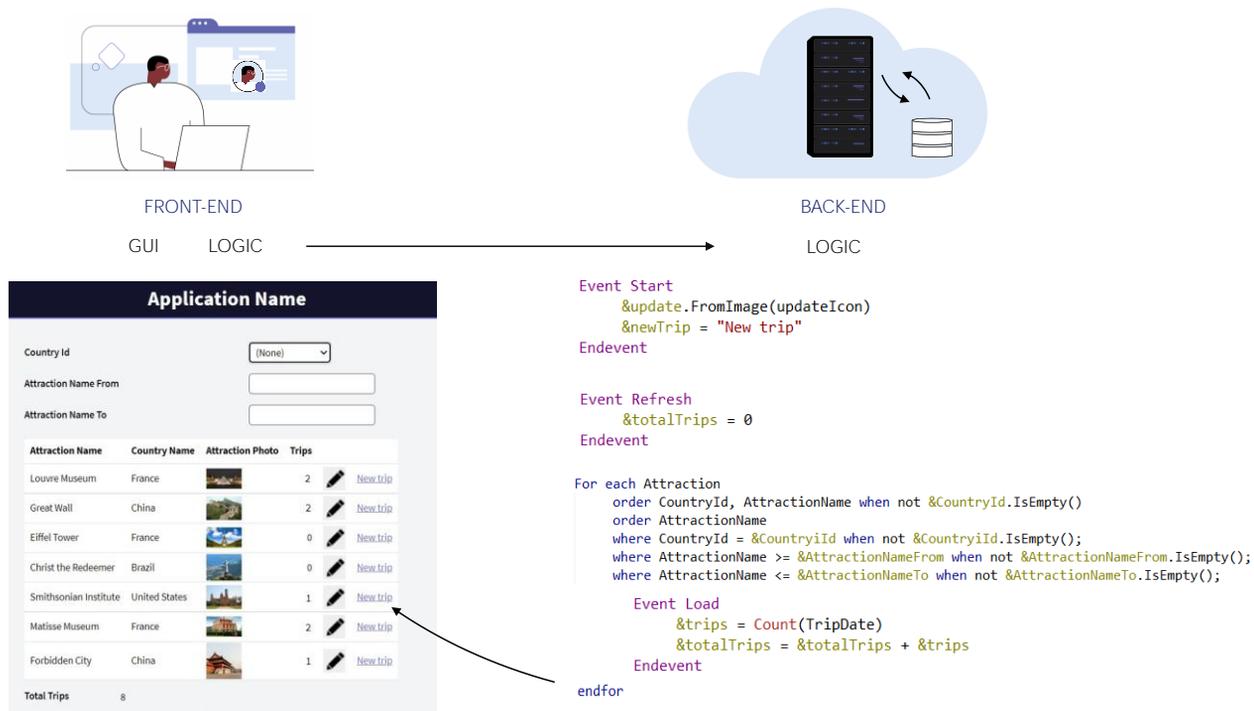


# Telas web com foco em Back-office

Objeto Web Panel. Esquema de execução de eventos

**GeneXus**<sup>™</sup>

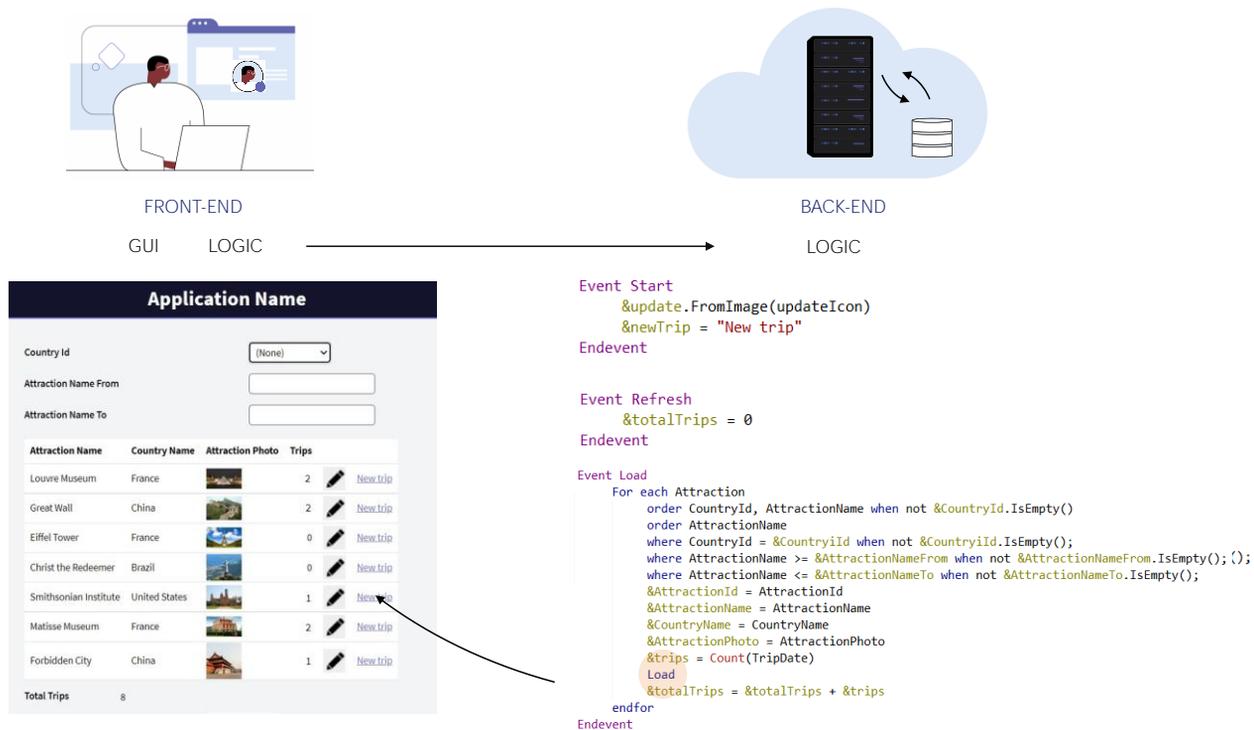
## Executando o Web Panel pela primeira vez



Se traduzirmos toda esta lógica que vimos para Front-end e Back-end interagindo, então...

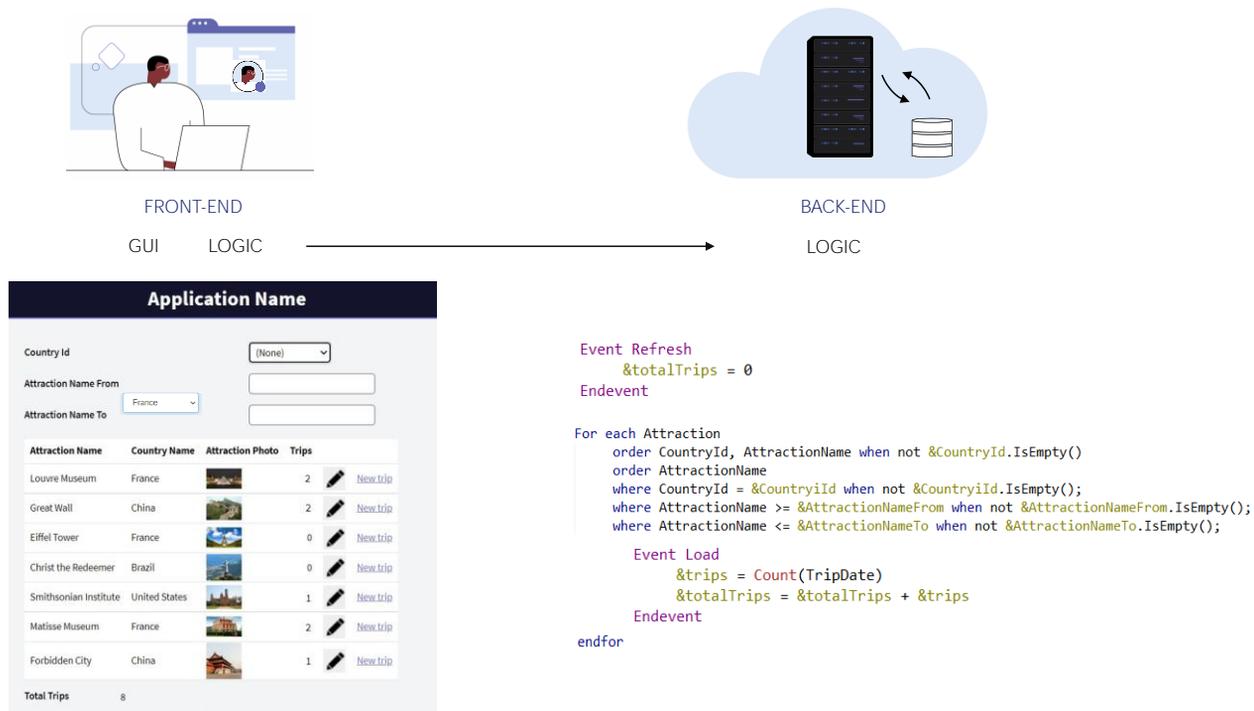
Quando se executa pela primeira vez o web panel, a partir da camada de lógica do Front-end é invocada a lógica do Web panel em Back-end, onde é executado: o evento Start, o evento Refresh e em seguida, se o grid tem tabela base, então dentro da lógica, GeneXus programou de maneira transparente um tipo de for each, para acessar a tabela base, respeitando order e conditions especificadas no grid, e para cada registro encontrado, executa o evento Load, adicionando, ao finalizar, uma linha para os dados a serem devolvidos ao Front-end. A adição da linha também é automática, o desenvolvedor não precisa especificar **comando** Load para isso. Ao finalizar, a resposta é enviada para o Front-end, que apresentará as informações na tela.

## Executando o Web Panel pela primeira vez



Em vez disso, se não houvesse tabela base, então esta parte de código não estaria, mas diretamente se dispararia o evento Load, uma única vez, onde se queremos acessar a base de dados, devemos programá-lo explicitamente, e é por isso que aqui aparece o comando for each programado pelo desenvolvedor. Para carregar uma linha no grid, o desenvolvedor deve indicar explicitamente com o comando Load, porque GeneXus não pode saber qual é a nossa intenção. Finalizado tudo isto, exatamente como no outro caso, a resposta é enviada para o Front-end, que apresentará as informações na tela.

## Executando o Web Panel pela primeira vez



Agora, o usuário irá interagir com a tela. Dependendo de qual seja a interação, o que acontecerá.

Por exemplo, se inserir um valor em uma das variáveis de filtro dos dados de um grid, a partir da camada de lógica do Front-end, será invocado o Back-end, passando-lhe os valores das variáveis. Ali, serão executados os eventos Refresh e Load para carregar novamente as informações desse grid. Novamente, se o grid tem tabela base, então este será o código que será executado no server....

## Mudanças nas variáveis dos filtros



**Application Name**

Country Id:

Attraction Name From:

Attraction Name To:

Attraction Name	Country Name	Attraction Photo	Trips
Louvre Museum	France		2 <a href="#">New trip</a>
Eiffel Tower	France		2 <a href="#">New trip</a>
Matisse Museum	France		2 <a href="#">New trip</a>

Total Trips: 5

```

Event Refresh
  &totalTrips = 0
Endevent

Event Load
  For each Attraction
    order CountryId, AttractionName when not &CountryId.IsEmpty()
    order AttractionName
    where CountryId = &CountryId when not &CountryId.IsEmpty();
    where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty();
    where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty();
    &AttractionId = AttractionId
    &AttractionName = AttractionName
    &CountryName = CountryName
    &AttractionPhoto = AttractionPhoto
    &trips = Count(TripDate)
    Load
    &totalTrips = &totalTrips + &trips
  endfor
Endevent

```

...e se não, será este outro.

Em qualquer caso, ao consultar novamente a base de dados, agora alguns registros não passarão pelo filtro. Mais uma vez, responderá ao Front-end que, agora, apresentará o grid com os dados resultantes, entre os quais está a variável &TotalTrips.

# Mudanças nas variáveis dos filtros



FRONT-END

GUI LOGIC



BACK-END

LOGIC

**Application Name**

Country Id:

Attraction Name From:

Attraction Name To:

Attraction Name	Country Name	Attraction Photo	Trips
Louvre Museum	France		2 <a href="#">New trip</a>
Eiffel Tower	France		2 <a href="#">New trip</a>
Matisse Museum	France		2 <a href="#">New trip</a>

Total Trips: 5 **Bold**

```
Event 'Bold'  
&totalTrips.FontBold = True  
Endevent
```

Se tivéssemos um evento cujo código pode ser resolvido no cliente como, por exemplo, mudar a fonte para negrito para um controle do form, esse código será executado no próprio Front-end, diretamente, sem necessidade de nenhuma viagem ao servidor.

## Mudanças nas variáveis dos filtros



FRONT-END

GUI    LOGIC



BACK-END

LOGIC

**Application Name**

Country Id:

Attraction Name From:

Attraction Name To:

Attraction Name	Country Name	Attraction Photo	Trips
Louvre Museum	France		2 <a href="#">New Trip</a>
Eiffel Tower	France		2 <a href="#">New Trip</a>
Matisse Museum	France		2 <a href="#">New Trip</a>

Total Trips: 5 **Bold**

```

Event &newTrip.Click
  &trips = NewTrip(AttractionId)
endevent

parm(in:&AttractionId, out:&trips);

new
  TripDate = Today()
  TripDescription = "Created automatically"
endnew
&tripId = TripId
new
  TripId = &tripId
  AttractionId = &AttractionId
endnew
&trips = Count(TripDate, AttractionId = &AttractionId)

```

Em vez disso, se necessita executar no servidor, como era o nosso exemplo, onde invocávamos um procedimento para criar um trip com a atração, então do Front-end é invocado para o Back-end, onde o evento associado se encontra programado, que invoca o procedimento que ali se executa, insere os registros, e devolve a execução ao evento que o chamou.

Neste caso, como se atribui o resultado do procedimento a uma variável do grid, então se envia ao Front-end a informação de que deve modificar o valor dessa linha, que é o que se faz ali.

## Mudanças nas variáveis dos filtros



FRONT-END

GUI    LOGIC



BACK-END

LOGIC

**Application Name**

Country Id:

Attraction Name From:

Attraction Name To:

Attraction Name	Country Name	Attraction Photo	Trips
Louvre Museum	France		2 <a href="#">New Trip</a>
Eiffel Tower	France		2 <a href="#">New Trip</a>
Matisse Museum	France		2 <a href="#">New Trip</a>

Total Trips: 5 **Bold**

```

Event &newTrip.Click
    &trips = NewTrip(AttractionId)
    Refresh
endevent

new
    TripDate = Today()
    TripDescription = "Created automatically"
endnew
&tripId = TripId
new
    TripId = &tripId
    AttractionId = &AttractionId
endnew
&trips = Count(TripDate, AttractionId = &AttractionId)

```

Mas, se depois da invocação encontrar um comando Refresh,

## Mudanças nas variáveis dos filtros



FRONT-END

GUI    LOGIC



BACK-END

LOGIC

**Application Name**

Country Id:

Attraction Name From:

Attraction Name To:

Attraction Name	Country Name	Attraction Photo	Trips
Louvre Museum	France		2
Eiffel Tower	France		2
Matisse Museum	France		2

Total Trips: 5 **Bold**

```

Event &newTrip.Click
    &trips = NewTrip(AttractionId)
    Refresh
endevent

Event Refresh
    &totalTrips = 0
Endevent

For each Attraction
    order CountryId, AttractionName when not &CountryId.IsEmpty()
    order AttractionName
    where CountryId = &CountryId when not &CountryId.IsEmpty();
    where AttractionName >= &AttractionNameFrom when not &AttractionNameFrom.IsEmpty();
    where AttractionName <= &AttractionNameTo when not &AttractionNameTo.IsEmpty();

    Event Load
        &trips = Count(TripDate)
        &totalTrips = &totalTrips + &trips
    Endevent
endfor

```

então antes de enviar uma resposta ao cliente, executa os eventos Refresh e Load, (aqui vemos o código para o grid com tabela base) após o que devolve ao Front-end todas as linhas do grid que cumprem as condições, novamente, e a variável &TotalTrips, que o usuário vê com os valores esperados.

*GeneXus*<sup>TM</sup>

[training.genexus.com](http://training.genexus.com)  
[wiki.genexus.com](http://wiki.genexus.com)