Pantallas web con foco en Customer-facing

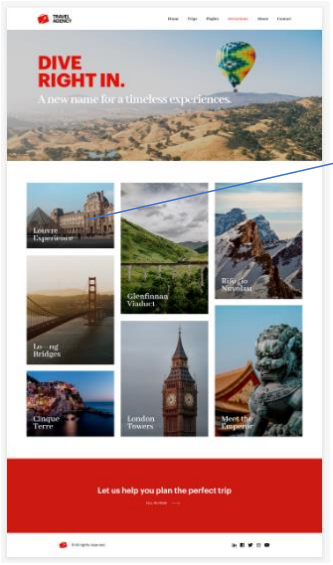
Primeros pasos con Angular

GeneXus™

Hasta ahora vimos cómo construir pantallas con foco en el back-office de la aplicación, es decir, en la parte que usan los empleados de la agencia de viajes para ingresar y mantener los datos de la empresa. Ahora veremos cómo diseñar e implementar las pantallas que utilizarán los clientes de la agencia para consultar información, lo que constituye la parte customer-facing de la aplicación.

Requerimiento de la Agencia: Aplicación Customer facing

Objetos Panel



La agencia de viajes nos pide que construyamos para sus clientes, una pantalla web que muestre las atracciones turísticas disponibles y poder interactuar con la información, por ejemplo filtrando los datos en pantalla para refinar una búsqueda o ver el detalle de una atracción seleccionada.

Antes vimos cómo construir pantallas con web panels, veremos ahora cómo implementarlas con el objeto panel, lo que nos permitirá generar la aplicación en Angular.

Recordemos que si quisiéramos, podríamos usar estos mismos objetos panel, para generar las pantallas de nuestra aplicación móvil, para dispositivos Android o Apple.

Desarrollando la aplicación

Prerrequisitos para Angular https://wiki.genexus.com/commwiki/servlet/wiki?

Vamos a crear la lista de atracciones disponibles, de forma de poder luego hacer clic sobre una que nos interese y ver más información de esa atracción. La idea es construir un panel similar al webpanel WWAttractionsFromScratch que vimos anteriormente

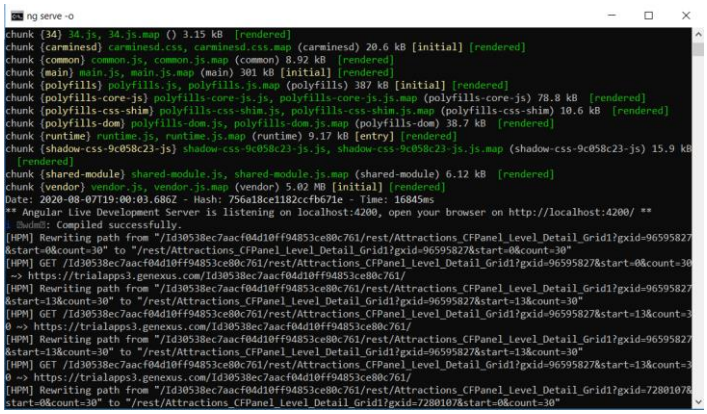
Creamos un folder FrontendAngular y creamos un objeto del tipo Panel de nombre Attractions_CFPanel. Como vemos, aquí también tenemos un formulario donde podemos arrastrar controles desde la barra de herramientas.
Comenzamos arrastrando un Grid y seleccionamos los atributos AttractionId, AttractionName, CountryName y AttractionPhoto. Luego a Attraction Id le ponemos la propiedad Visible en False y en la propiedad Base Trn del grid, asignamos la transacción Attraction.

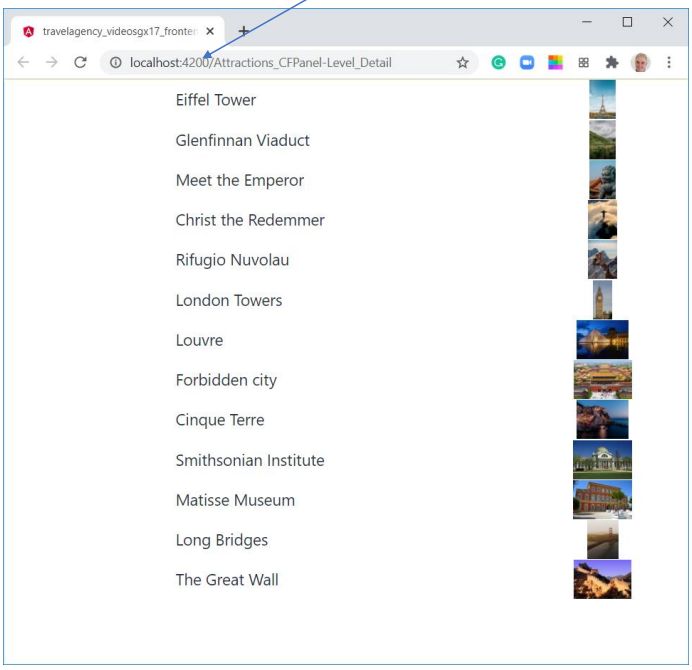
Ahora asignaremos a Angular como generador. En el KB Explorer hacemos clic en Front end y observamos que hay una propiedad llamada Generate Angular que su valor por defecto es False, así que la ponemos en True. También vemos que Generate Android y Generate Apple están por defecto en True, pero como no nos interesa por ahora generar para dispositivos móviles, las ponemos en False.

Para poder generar en Angular, debemos instalar el software que se menciona en el artículo del wiki, que se muestra en pantalla.

Ejecución de la aplicación

El valor del puerto puede cambiar y se asignará

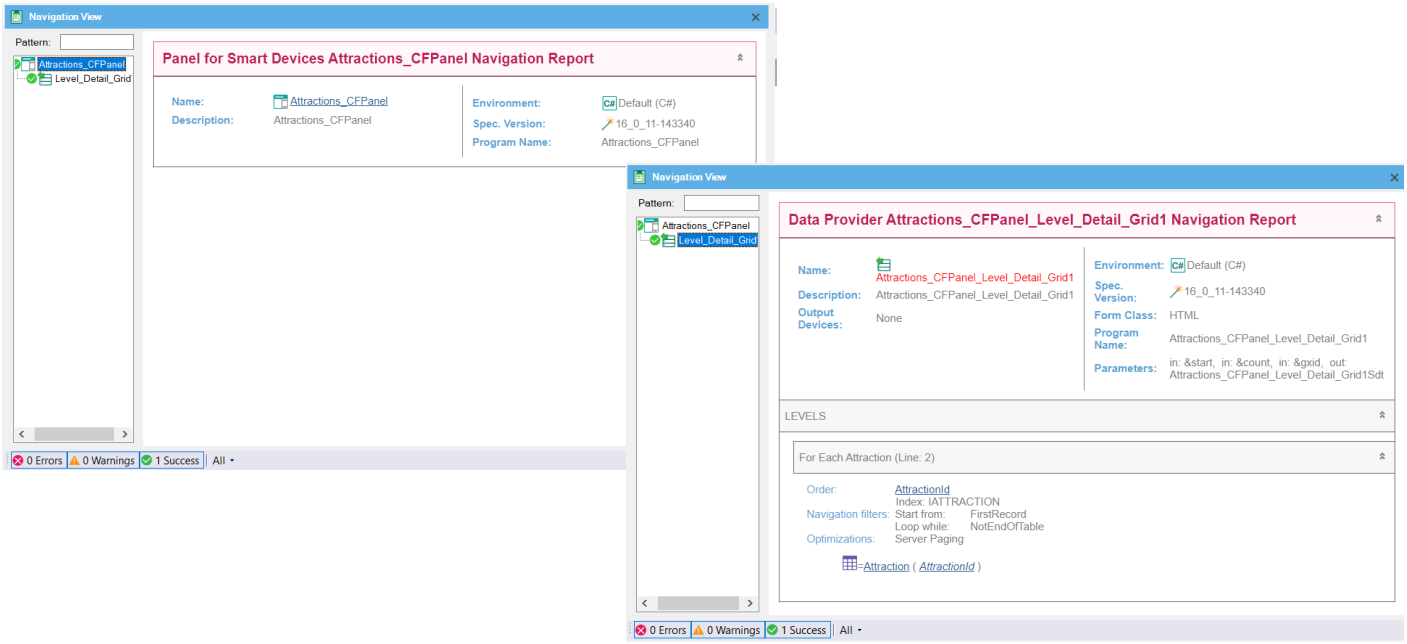




Para ejecutar, primero ponemos al panel que acabamos de crear como Main, damos botón derecho sobre él y seleccionamos Run. Vemos que se abre una ventana de línea de comandos que muestra la ejecución del servidor y luego de unos minutos se abre el navegador con la aplicación en ejecución.

Obviamente el diseño deja bastante que desear, pero ya nos ocuparemos de eso. La aplicación se está ejecutando por defecto en la url : http://localhost:4200, que corresponde a la dirección del servidor local para desarrollo, donde la aplicación se instaló automáticamente.
Ya tenemos funcionando nuestra primera aplicación en Angular.

Listados de navegación



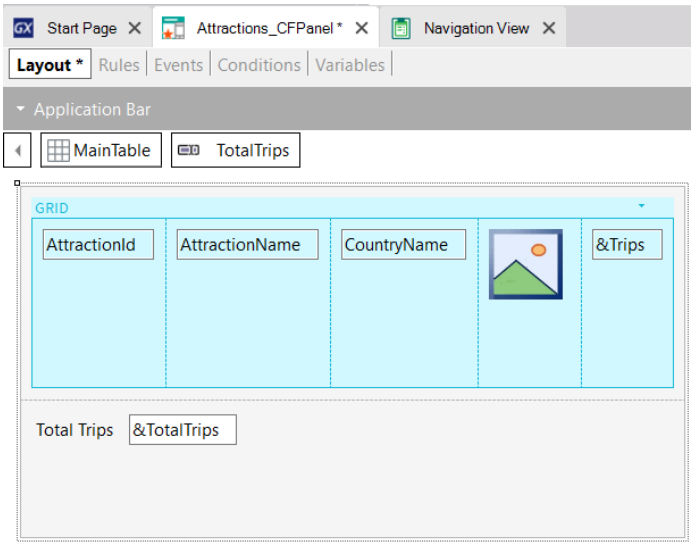
Si vamos al listado de navegación del objeto panel Attractions_CFPanel vemos que aparecen dos entradas. Si seleccionamos la que tiene el nombre del panel, el listado está vacío y no aparece ninguna referencia a la tabla Attraction. Eso está en el nodo llamado Level_Detail_Grid1, donde vemos que la información es similar a la que veríamos con un web panel con tabla base Attraction, ya que la grilla del panel está recorriendo la tabla Attraction para mostrar las atracciones turísiticas.

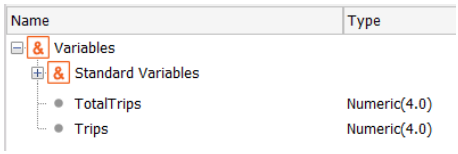
El nodo del listado de nombre Attractions_CFPanel corresponde al panel mismo, e incluye información de elementos de User Interface, como por ejemplo la navegación de los Dynamic combo. Como en nuestro panel no tenemos ningún elemento en la parte fija, el listado aparece vacío.

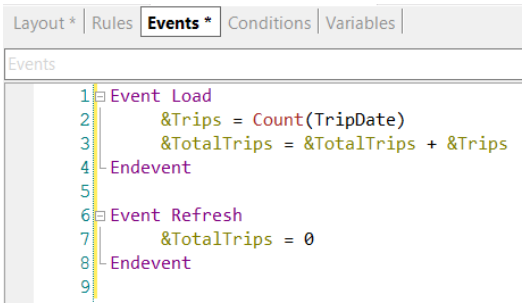
El nodo llamado nivel Level_Detail_Grid1 corresponde a la grilla que insertamos. Vemos que se recorre la tabla base Attraction tal como esperábamos, al haber insertado atributos de Attraction en el grid. En el título del reporte dice que la navegación corresponde al Data Provider Attractions_CFPanel_Level_Detail_Grid1, que es el data provider publicado como servicio en el back-end e invocado por el panel para acceder a la base de datos y recuperar las atracciones.

Aquí tenemos la prueba de lo que vimos antes, que la arquitectura de estas aplicaciones customer-facing con Angular, tienen la lógica a nivel del cliente y se invocan servicios en el servidor para acceder a la información de la base de datos.

Agregamos total de viajes de cada atracción y total global de viajes







Para que nuestro panel quede más parecido al webpanel de Trabajar con Atracciones visto antes, vamos a agregar una columna más al grid con el total de viajes de cada atracción y fuera del grid, el total de viajes de todas las atracciones. Creamos las variables &Trips y &TotalTrips, agregamos a &Trips en la grilla poniendo su propiedad Label position en None y a &TotalTrips abajo del grid.

Para obtener la cantidad de viajes de una atracción, hay que recorrer la tabla TripAttraction. Como esto está relacionado a la base de datos, tenemos que programar un evento que se dispare en el servidor. En los objetos panels también contamos con los eventos Start, Refresh y Load al igual que en los webpanels.

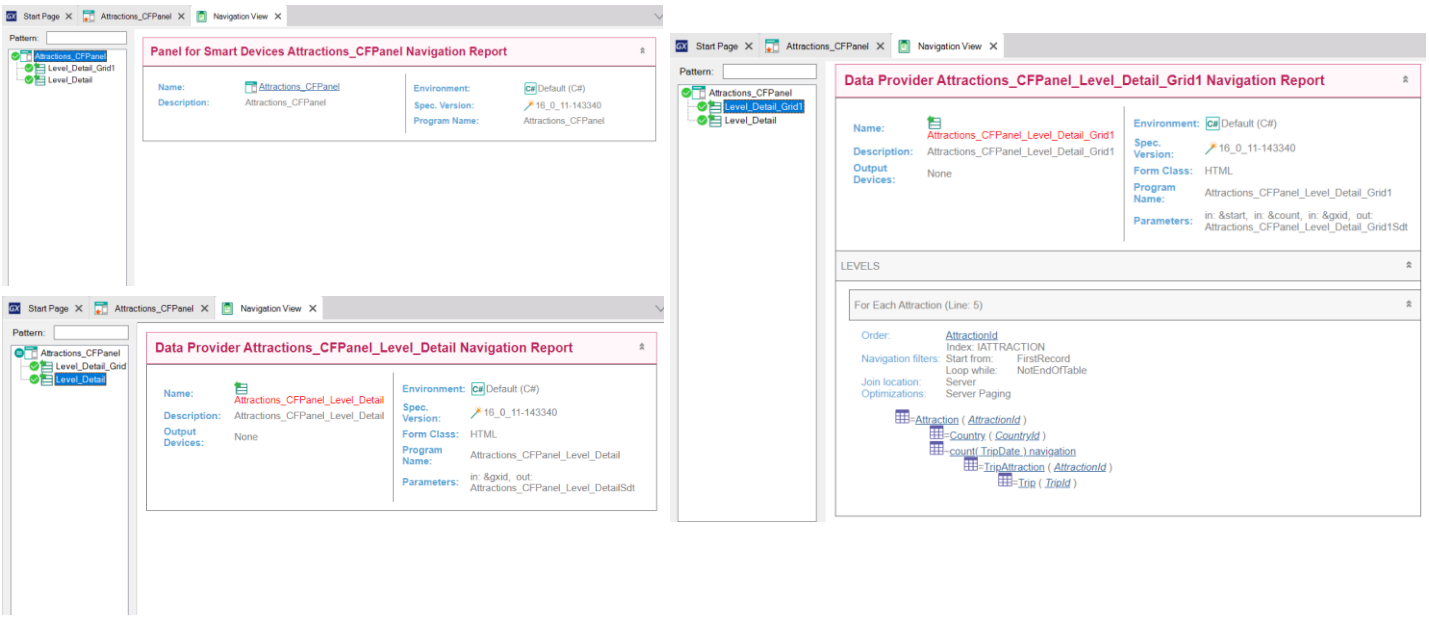
Vamos a programar los eventos igual que como hicimos con el web panel WWAttractionsFromScratch. En el evento Load usamos una fórmula Count que mediante el atributo TripDate cuente los registros de los viajes, ya como vimos antes, si bien el atributo TripDate es de la tabla Trip, GeneXus no escogerá la tabla Trip como tabla base de la fórmula, sino la tabla TripAttraction.
Como sabemos que la grilla recorre la tabla Attraction porque lo vimos en el listado de navegación, el evento Load se disparará una vez por cada renglón de la grilla, por lo que la fórmula count contará los viajes de cada atracción mostrada. Luego acumulamos en &TotalTrips el total de todos los viajes.
Cabe la misma aclaración que vimos antes, en este caso usamos el evento Load general porque tenemos un solo grid, pero

podríamos haber usado el evento Grid1.Load, lo que nos permite que en el futuro podamos agregar otro grid al panel y no nos cambie la programación.

Como queremos que el total de viajes se inicialice en cero antes de empezar a contar los viajes de las atracciones, vamos a agregar en el evento Refresh la inicialización a la variable &TotalTrips, en forma análoga a lo que habíamos hecho en el webpanel.

Vamos a ejecutar nuevamente, así que damos botón derecho sobre el panel y seleccionamos Run.

Listados de navegación



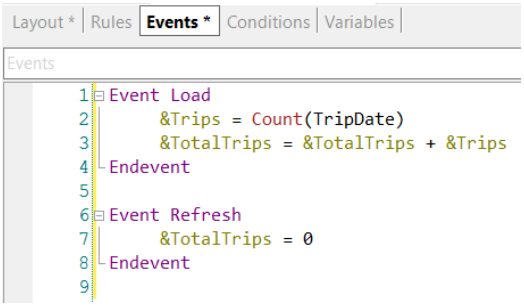
Si analizamos el listado de navegación, vemos que aparece un nuevo nivel de detalle distinto al del grid, llamado Level_Detail. Aquí es donde se muestra la carga de la parte fija del panel, es decir de todos los controles del form que no están incluidos en un grid.

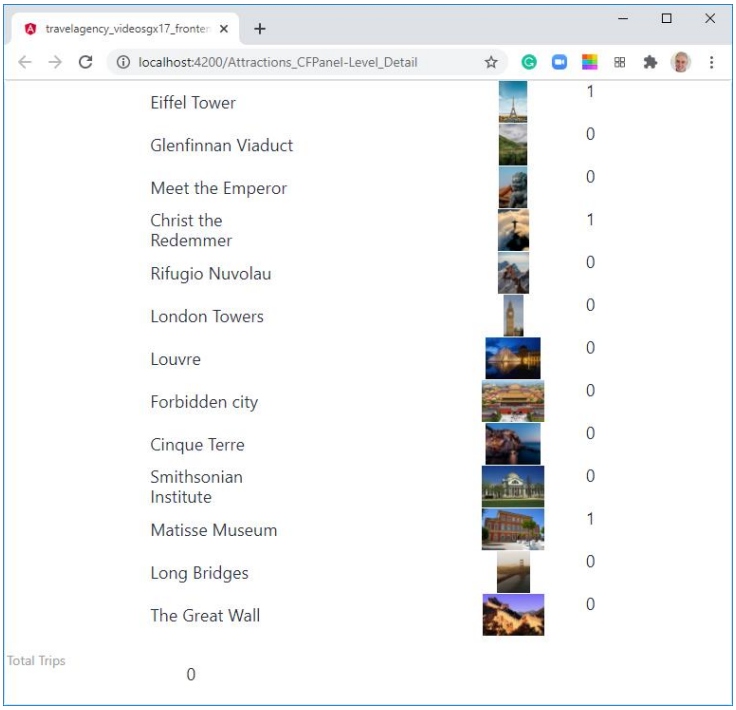
A diferencia de los web panels, en un objeto panel la parte fija es independiente del grid e incluso, como veremos más adelante, la parte fija puede tener una tabla base distinta a la tabla base del grid.

El listado de navegación del nodo Level_Detail (que invoca al data provider mencionado en el título) aparece vacío, porque no se está cargando ningún campo desde la base de datos, solamente está la variable &TotalTrips, que se actualiza dentro del evento Load.

Si vamos al listado de navegación del Level_Detail_Grid1, vemos que el data provider correspondiente está accediendo a la tabla Attraction, ordenada por AtractionId que es el orden por defecto y que también aparece la navegación de la fórmula Count ya que la misma se dispara en el evento Load.

Total global de viajes mal calculado!





Si vamos a la aplicación ejecutando en el navegador, vemos que aparece el total de viajes de cada atracción en forma correcta, pero que el total acumulado de viajes sale en 0.
¿Qué hicimos mal? La variable &TotalTrips se está incrementando en el evento Load como hicimos en el webpanel y tenemos la seguridad de que este evento se está disparando porque vemos que algunas atracciones tienen viajes.... ¿Entonces?

La razón es que los objetos panels no funcionan igual que los web panels. En los objetos panel, la parte fija se carga en forma independiente de la grilla.
En este caso, la variable &TotalTrips está en la parte fija del panel, que es lo que primero se carga y luego se produce la carga del grid, por lo tanto cuando se muestra la variable aún no hay podido cargarse el grid, no se ha disparado aún el evento Load y no se ha podido acumular el valor de &TotalTrips.

Recordemos que en un objeto panel usado para desarrollar aplicaciones con foco en customer-facing, para cargar la pantalla en el dispositivo cliente, se invoca a servicios ubicados en el servidor que son los que acceden a la base de datos. Estos servicios son data providers, que son independientes para la parte fija y para el grid (o cada grid) de la pantalla.

Cuando se inicia la ejecución del panel, se dispara un evento local en el cliente que invoca al data provider para cargar la parte fija y hace que se disparen los eventos Start y Refresh en el servidor. Luego se ejecuta un segundo data provider que dispara el evento Load en el servidor N veces y se carga la grilla.

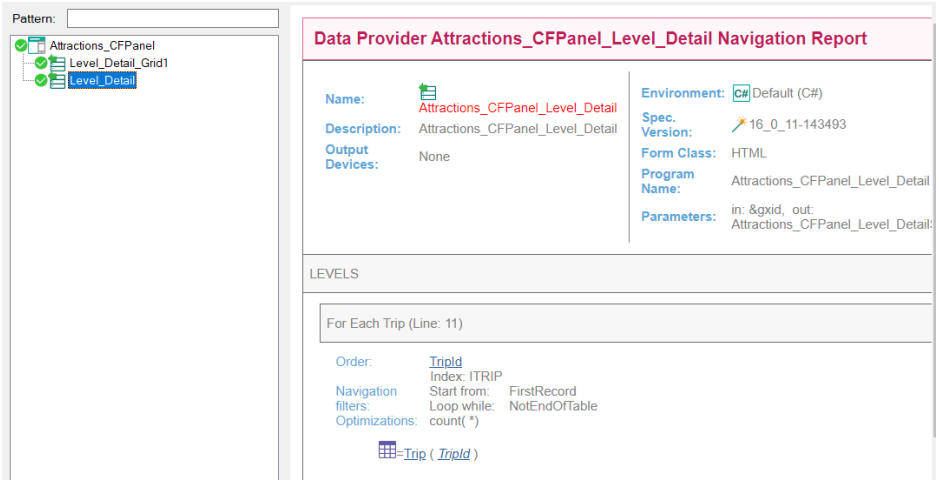
En nuestro ejemplo, al dispararse el Refresh primero, se inicializa la variable &TotalTrips y se carga la parte fija, recién después se dispara el evento Load que es donde &TotalTrips se carga con el valor correcto y se actualiza el grid, pero la parte fija ya se cargó antes y no se vuelve a dibujar.

Debido a esta característica no podemos programar el objeto panel como si fuera un webpanel.

En otro video entraremos en detalle en el disparo de los eventos y en la determinación de las tablas base, pero por ahora, para que nos funcione el ejemplo, vamos a cambiar la programación.

Solución correcta

```
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
```



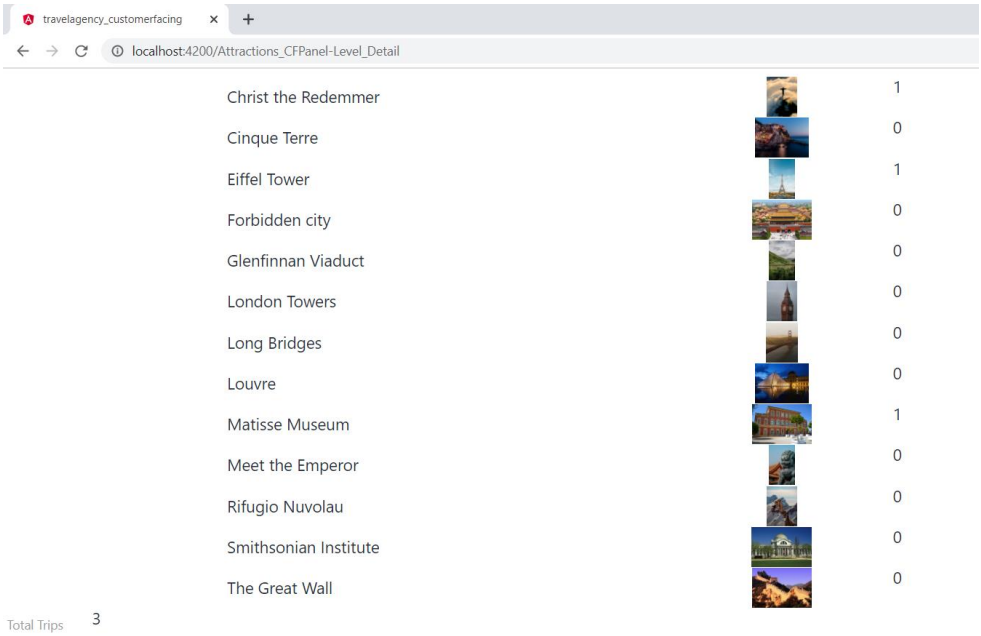
La solución es incluir en el evento Refresh un For Each que acceda a la tabla TripAttraction y cuente el total global de viajes. No nos olvidemos de eliminar el cálculo que teníamos en el evento Load.

La razón por la cual incluimos la actualización en el evento Refresh, utilizando un For Each, es porque el evento Refresh se disparará cuando se cargue la parte fija, que va a ser antes de cuando se cargue el grid.
Por lo tanto al cargar la parte fija el valor de &TotalTrips tendrá el valor correcto y recién después se actualizará la parte del grid.

Ejecutamos...

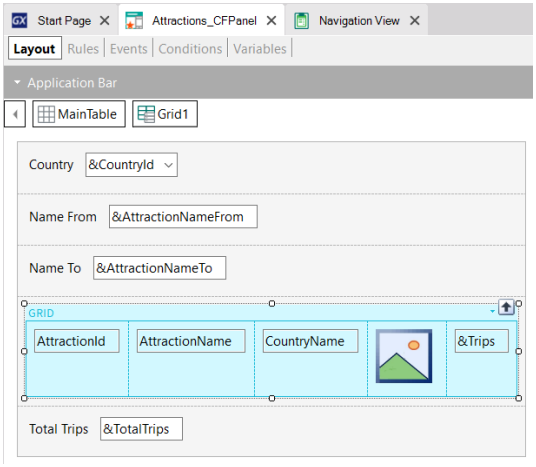
Observamos que ahora el listado de navegación correspondiente al nodo Level_Detail, incluye el For Each con la navegación a la tabla TripAttraction.

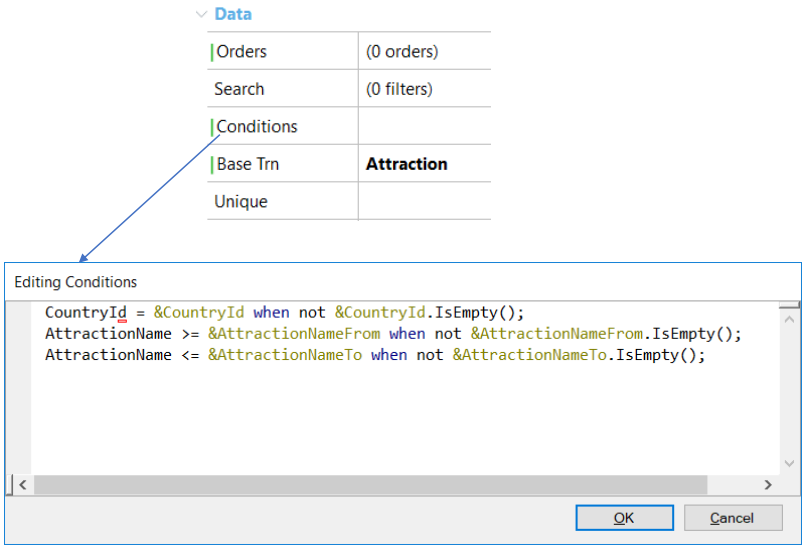
Ejecución con el total acumulado de viajes correcto



En el navegador vemos que ahora se muestra correctamente el total acumulado de viajes.

Agregamos filtros por país y nombre de atracción



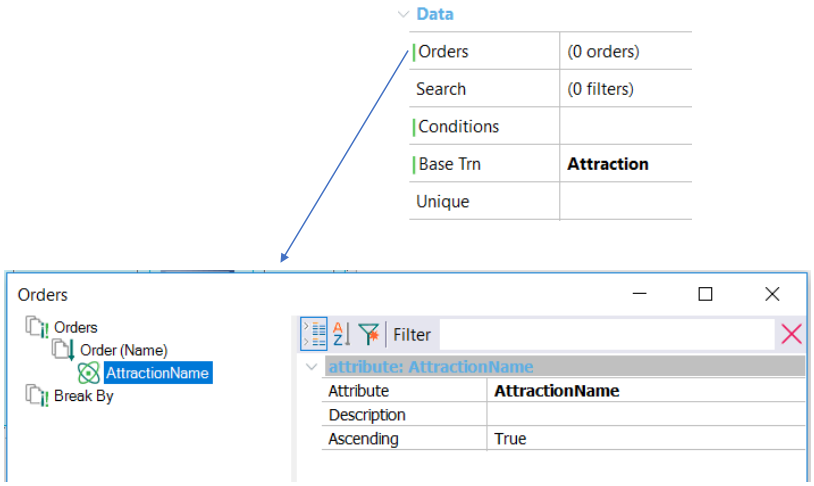


Para completar el panel de trabajar con atracciones, nos faltaría agregar los filtros por el identificador de país y por nombres de atracción.

Agregamos arriba del grid una variable &CountryId como Dynamic Combobox, con la propiedad Item Descriptions que muestre el CountryName y seteamos la propiedad Empty Item en True. Agregamos también las variables &AttactionNameFrom y &AttractionNameTo para filtrar las atracciones por nombre.

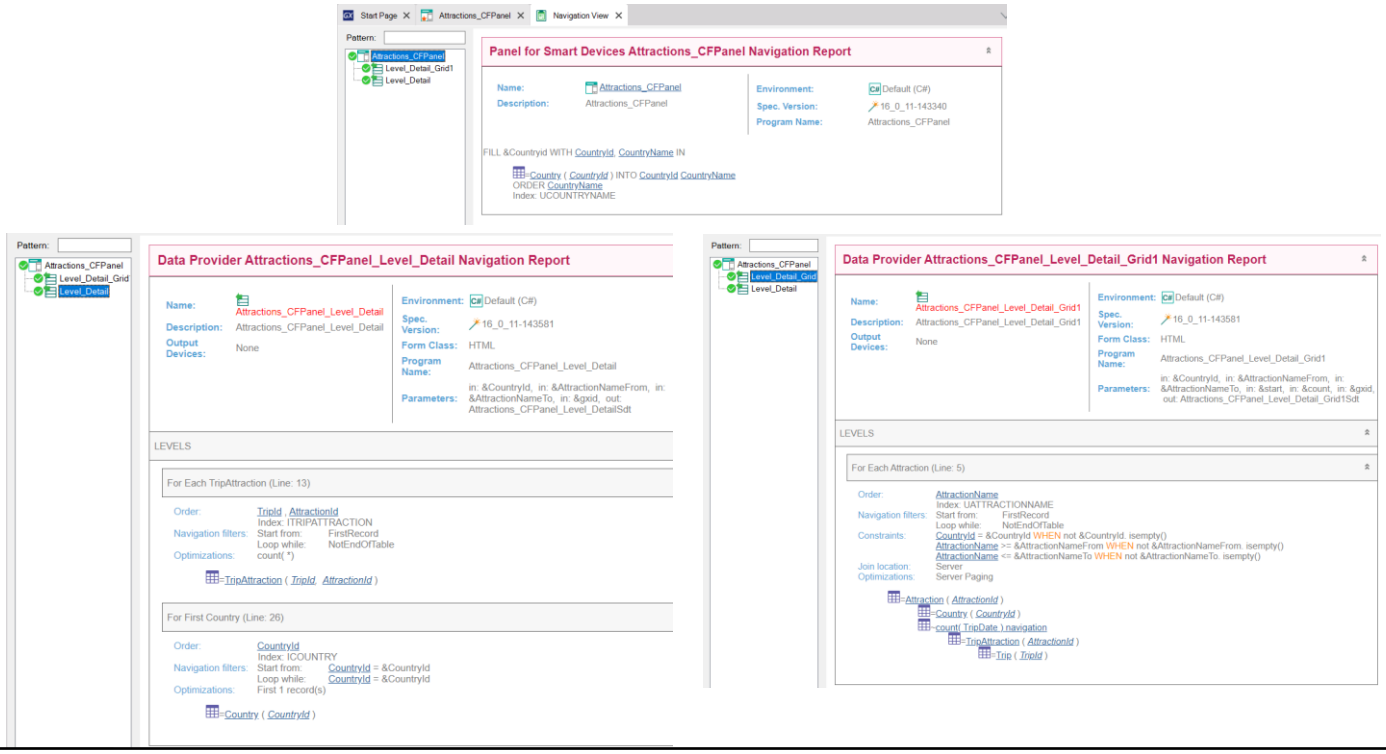
En un webpanel agregaríamos el filtro en las conditions del grid, aquí también podemos hacer lo mismo. Definamos una condition para filtrar por país y otras para filtrar desde y hasta el nombre de atracción.

Ordenamos la lista de atracciones por nombre



También disponemos de una propiedad Order, donde vamos a indicar que la grilla salga ordenada por nombre de atracción, así que, damos botón derecho sobre Orders, le damos un nombre, por ejemplo Name y luego damos otra vez botón derecho para ingresar al atributo AttractionName.

Podemos definir un orden por varios atributos, así como también crear otros órdenes por distintos criterios. La parte donde dice Break By, nos permite agrupar los registros del grid, por ejemplo si quisiéramos que las atracciones salieran agrupadas por nombre de país.

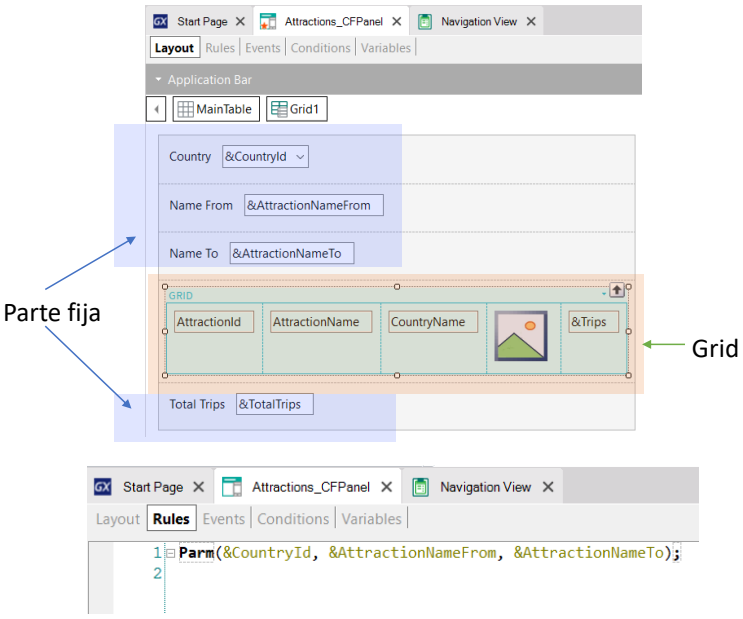


Si damos botón derecho sobre el panel y seleccionamos View Navigation, en el listado de navegación del panel mismo, vemos que se accede a la tabla Country para llenar el Dynamic Combobox (donde dice FILL &CountryId WITH CountryId, CountryName IN).

El listado de navegación del nodo Level_Detail nos muestra el reporte del Data Provider (creado automáticamente por GeneXus pero que no se ve en la KB) que es invocado en el servidor para recuperar los datos de la base de datos, necesarios para cargar la parte fija del panel. Éste data provider dispara en el servidor el evento Start y el Refresh.
En el listado vemos el For each que programamos en el evento Refresh, que accede a la tabla TripAttraction para contar el total de viajes y vemos también el acceso a la tabla Country filtrada por el CountryId seleccionado en el Dynamic combo.

Si vemos el listado de navegación correspondiente al data provider que carga el grid, vemos que ahora las atracciones se recorrerán ordenadas por AttractionName y que en las constraints aparecen los filtros que definimos. Este data provider ejecuta internamente el evento Load una vez por cada línea del grid a ser cargada al igual que en los webpanels, si el grid tiene tabla base y devuelve la información al panel para ser mostrada.

Necesidad de refrescar el contenido con los filtros




```
Event Load
      &Trips = Count(TripDate)
Endevent

Event Refresh
      &TotalTrips = 0
      For each Trip.Attraction
            &TotalTrips += 1
      Endfor
Endevent

Event &CountryId.ControlValueChanged
      Grid1.Refresh()
Endevent

Event &AttractionNameFrom.ControlValueChanged
      Grid1.Refresh()
Endevent


Event &AttractionNameTo.ControlValueChanged
      Grid1.Refresh()
Endevent
```

Una de las cosas que mencionamos fue que la parte fija del panel se carga en forma independiente de la carga del grid, invocándose data providers diferentes, que están publicados en el servidor como servicios y acceden a la base de datos para recuperar la información de cada parte.

Cuando cambiamos el valor de un filtro, es necesario que se refresque la información del grid. Para esto, es necesario que agreguemos el método Refresh del grid, que disparará los eventos Refresh y Load del servidor, lo que hará que se cargue nuevamente el grid, aplicando las conditions programadas y muestre los resultados filtrados como esperamos.

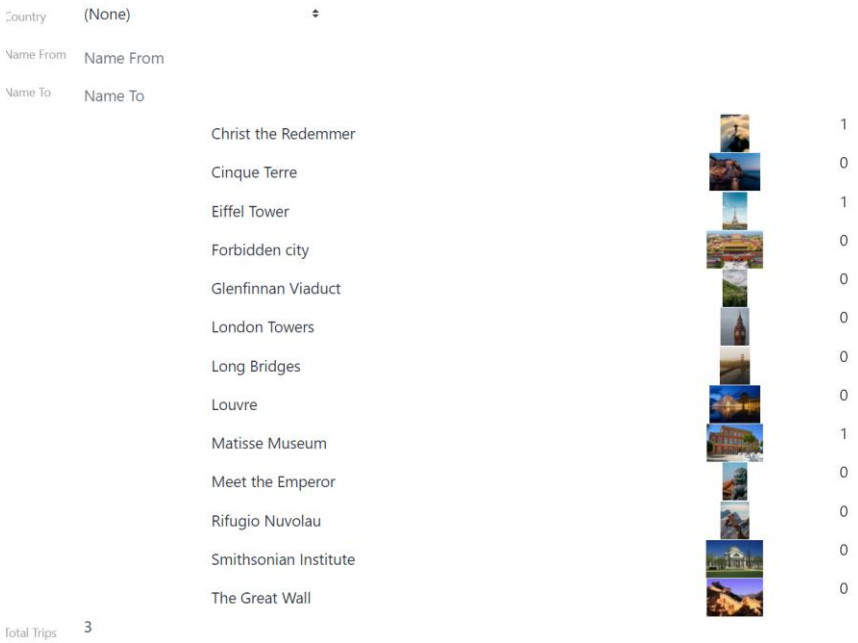
Como el método Refresh del grid lo debemos invocar luego de que cambiamos el valor de la variable del filtro, usamos el evento ControlValueChanged de cada variable, para invocar al método. De esta forma, luego de cambiar un valor, al salir del campo se disparará el evento correspondiente que terminará refrescando el contenido del grid.

Sin embargo, hay otra cosa que debemos tomar en cuenta y es que en esta arquitectura, como el objetivo es que la página se cargue las menor cantidad de veces posible, se prioriza el caché de datos, es decir que se trata siempre de recuperar la información previamente almacenada. Para que el servidor entienda que queremos traer nuevos datos, hay que hacer que se cambie la URL enviada al servidor, de forma que éste interprete que es una página nueva y obtenga la información correspondiente para enviarla al cliente.

Para eso, agregamos una regla Parm, que contenga los valores de las variables que usamos en los filtros, de forma que si cambia el valor de una variable, se actualice la página con los nuevos datos.

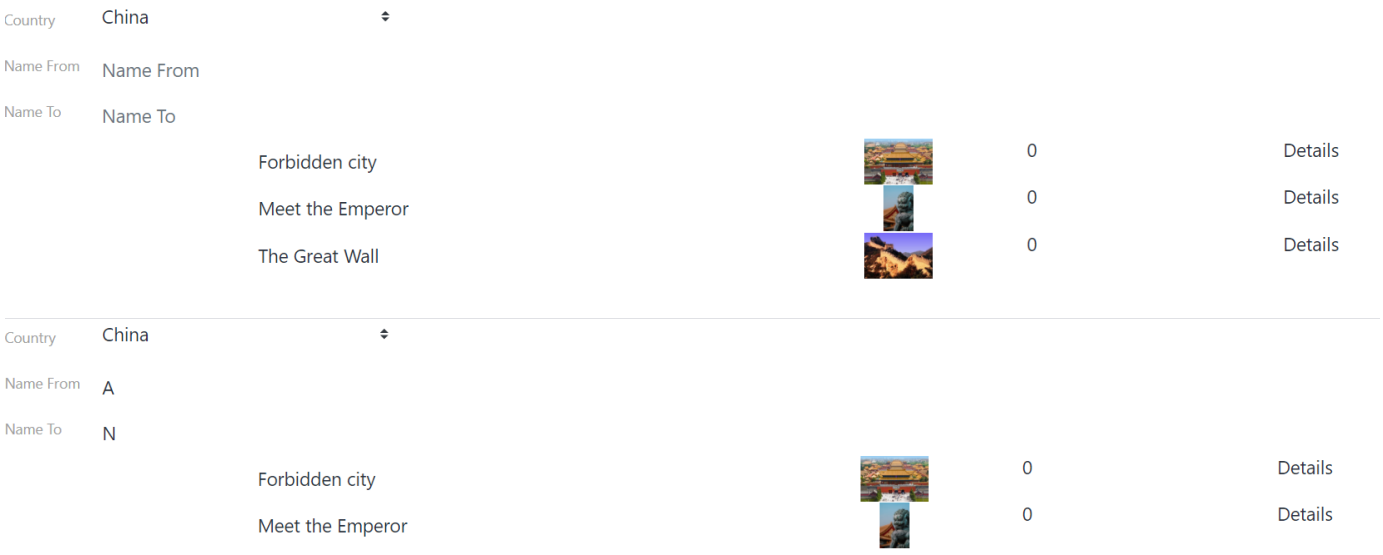
Ahora sí, ejecutamos para probar todo esto que vimos.

Ejecución con los nuevos filtros y orden



Vemos que en la parte superior ahora aparecen los filtros que agregamos y las atracciones están ordenadas por nombre como esperábamos.

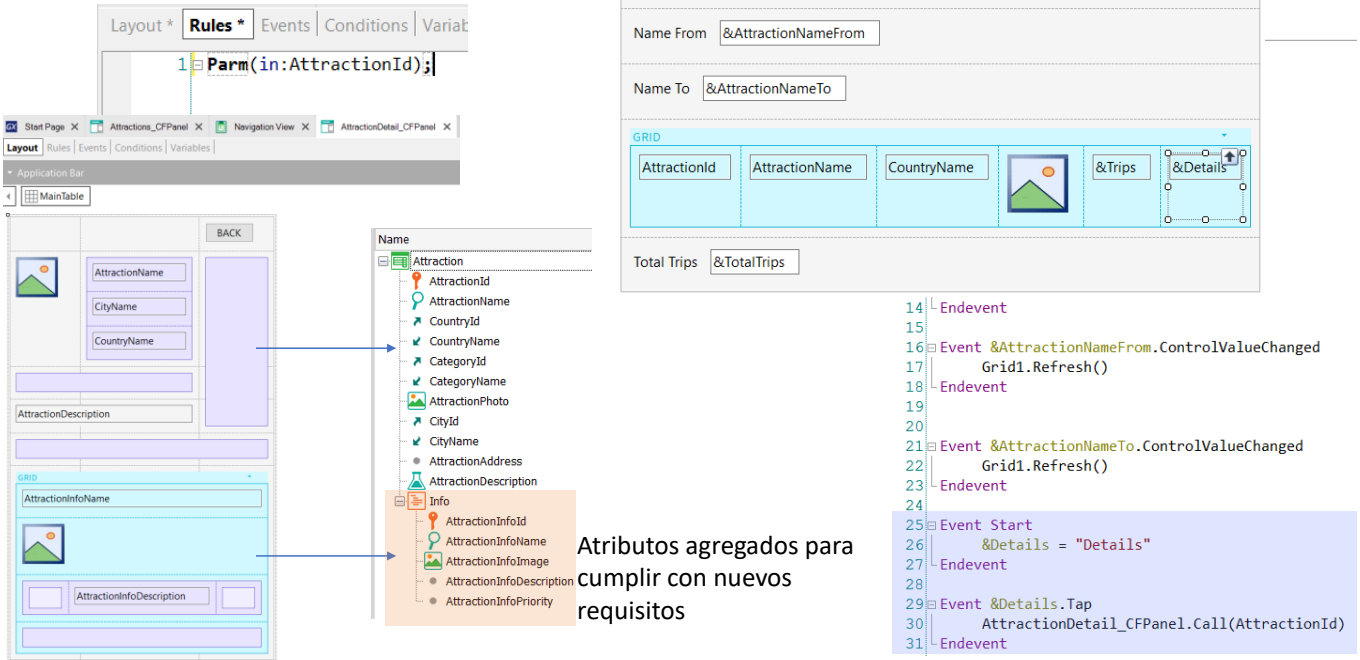
Ejecución con los nuevos filtros y orden (cont.)



Vamos a filtrar por el país China y vemos que nos muestra las atracciones de China.

Ahora elegimos ver las atracciones que comienzan con la letra A hasta la letra N y solamente vemos a las atracciones de China con el nombre en ese rango.

Implementamos el detalle de una atracción



Atributos agregados para cumplir con nuevos requisitos

```
14  Endevent
15
16  Event &AttractionNameFrom.ControlValueChanged
17      Grid1.Refresh()
18  Endevent
19
20
21  Event &AttractionNameTo.ControlValueChanged
22      Grid1.Refresh()
23  Endevent
24
25  Event Start
26      &Details = "Details"
27  Endevent
28
29  Event &Details.Tap
30      AttractionDetail_CFPanel.Call(AttractionId)
31  Endevent
```

Ahora completemos el requerimiento pedido por la agencia, que al hacer clic sobre una de las atracciones listadas, se muestre el detalle de la misma. Para eso utilizaremos otro objeto panel de nombre AttractionDetail_CFPanel. Para ahorrar tiempo, yo ya lo tengo creado.

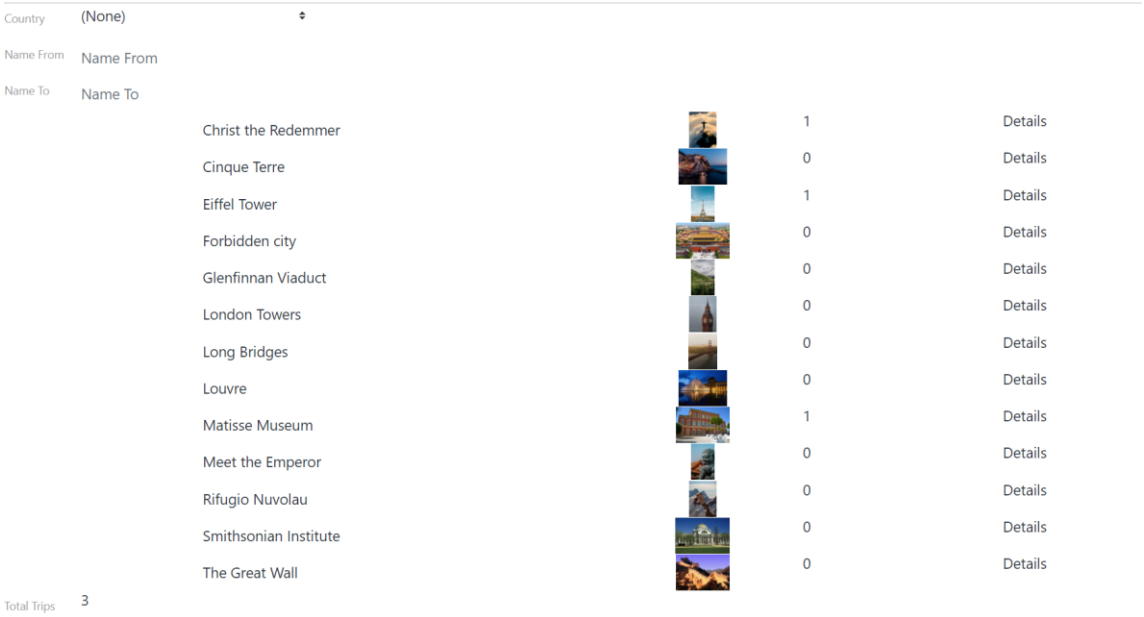
Vemos que en las reglas definimos una regla Parm, con un parámetro de entrada, el atributo AttractionId.
Recordemos que esto permitirá que se muestre solamente la información de la atracción que pasemos por parámetro, que fue la que seleccionamos en el panel de la lista de atracciones.

En el form pusimos al atributo AtractionPhoto y a los atributos AtractionName, CityName y CountryName. Más abajo insertamos al atributo AttractionDescription. Luego insertamos un grid y seleccionamos los atributos AttractionsInfoName, AttractionInfoImage y AttractionInfoDescription para ver la información de la atracción. Como vemos hemos insertado unos controles Table para alinear mejor.
También agregamos arriba a la derecha, un botón BACK que invocará a un Return que nos permitirá volver al listado de atracciones.

Ahora vamos al panel Attractions_CFPanel y agregamos una variable &Detail del tipo Character al grid. En evento Start le asignamos el texto “Details”.
Luego damos botón derecho sobre la variable del grid y seleccionamos Go to Event, Tap y escribimos la invocación a AttractionDetail_CFPanel, pasándole como parámetro AttractionId.

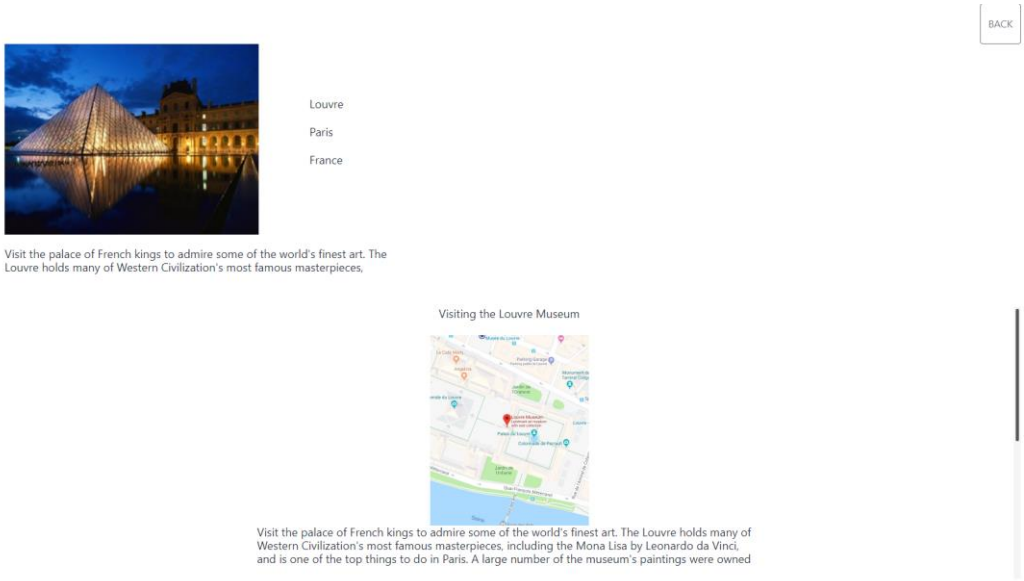
Probemos esto en ejecución.

Ejecución con el detalle de una atracción



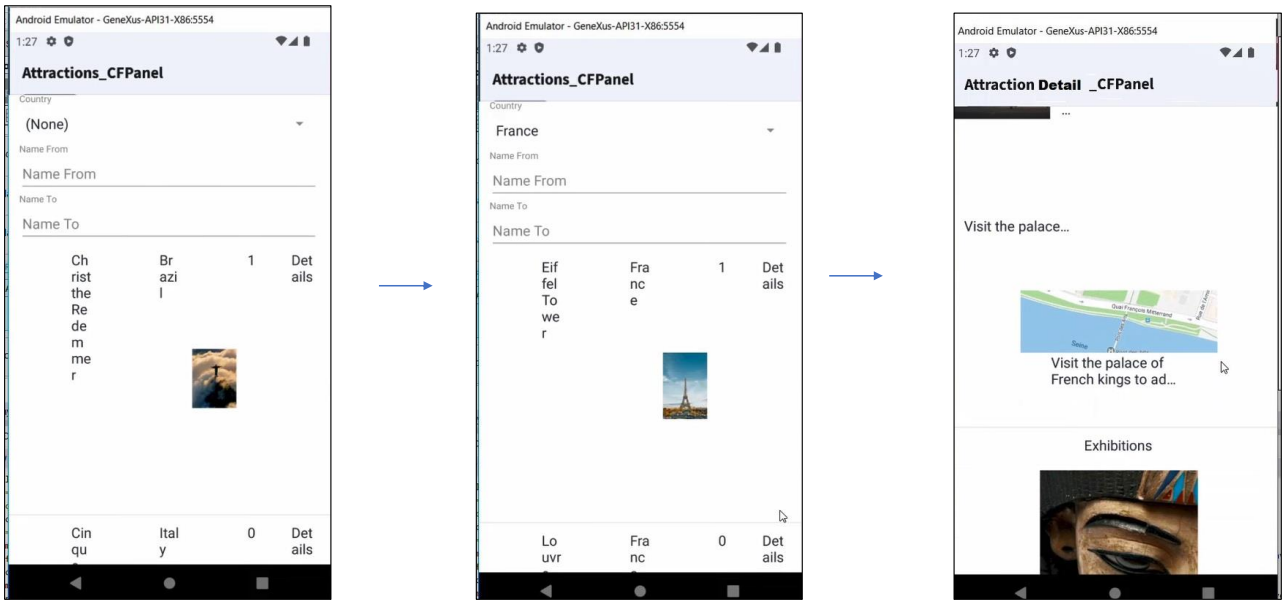
Vamos a ver la información del museo del Louvre, así que en el renglón correspondiente hacemos clic en Details.

Ejecución con el detalle de una atracción (cont.)



Se abre el panel con el detalle de la atracción, donde podemos apreciar un mapa y las exhibiciones disponibles.

Prerrequisitos para Android: https://wiki.genexus.com/commwiki/servlet/wiki?14449



Ahora bien... Al principio de este video dijimos que si quisiéramos, podríamos usar los objetos panel que construimos para generar las pantallas de nuestra aplicación móvil. Vamos a probar esto.

Para poder generar en lenguaje Android, deberá instalar el software que se menciona en el artículo del wiki, que se muestra en pantalla.

En el KB Explorer hacemos clic sobre el nodo Front end y ponemos la propiedad Generate Android en True. Ahora ejecutamos nuestro objeto main Attractions_CFPanel.

Vemos que se abre un emulador de Android mostrando el panel Attractions_CFPanel, con la lista de atracciones turísticas.
Obviamente no se ve muy bien, ya que cuando implementamos este panel estábamos pensando en un sistema web, que correría en la pantalla de un notebook o un computador de escritorio y deberíamos definir un diseño acorde al tamaño de pantalla de un teléfono.

Pero más allá del diseño, vamos a verificar si funciona correctamente. En el filtro elegimos France y vemos que se muestra solamente las atracciones de Francia. Ahora hacemos clic en Details del Louvre y vemos que se abre la pantalla con los detalles del museo, donde vemos el mapa y las exhibiciones como esperábamos.

En este video comenzamos a familiarizarnos con los objetos panel, generando la aplicación en el framework Angular y comprobamos que con los mismos objetos creados pudimos generar también la aplicación en lenguaje nativo Android.

A continuación conoceremos más del manejo de eventos a nivel del cliente y del servidor.

GeneXus™

training.genexus.com
wiki.genexus.com