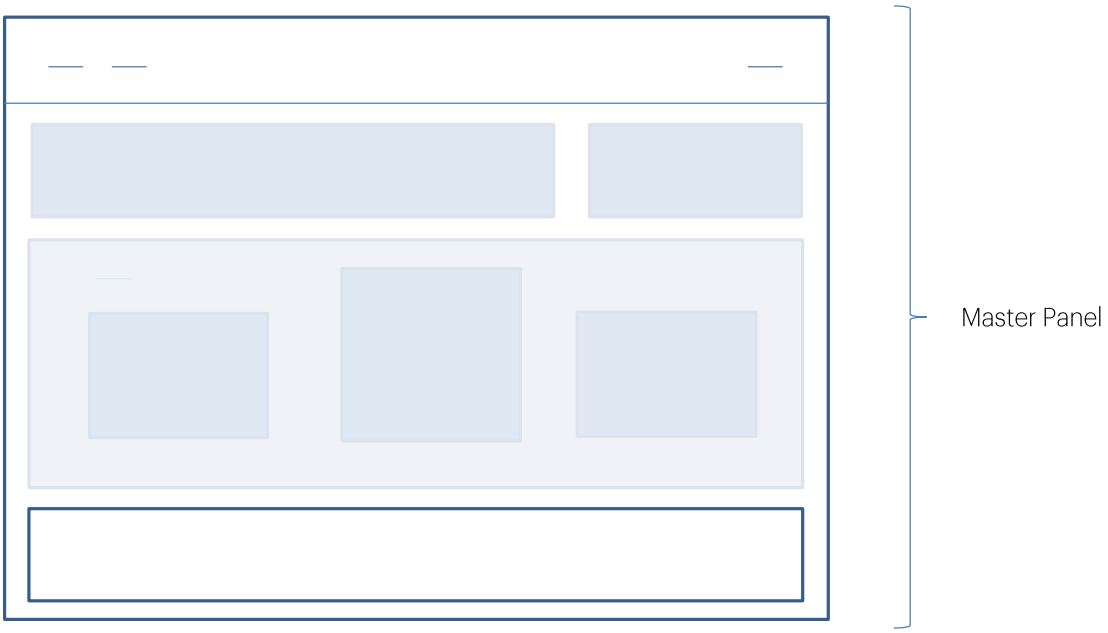
Eventos globales

Interacción entre componentes de una misma pantalla

GeneXus™

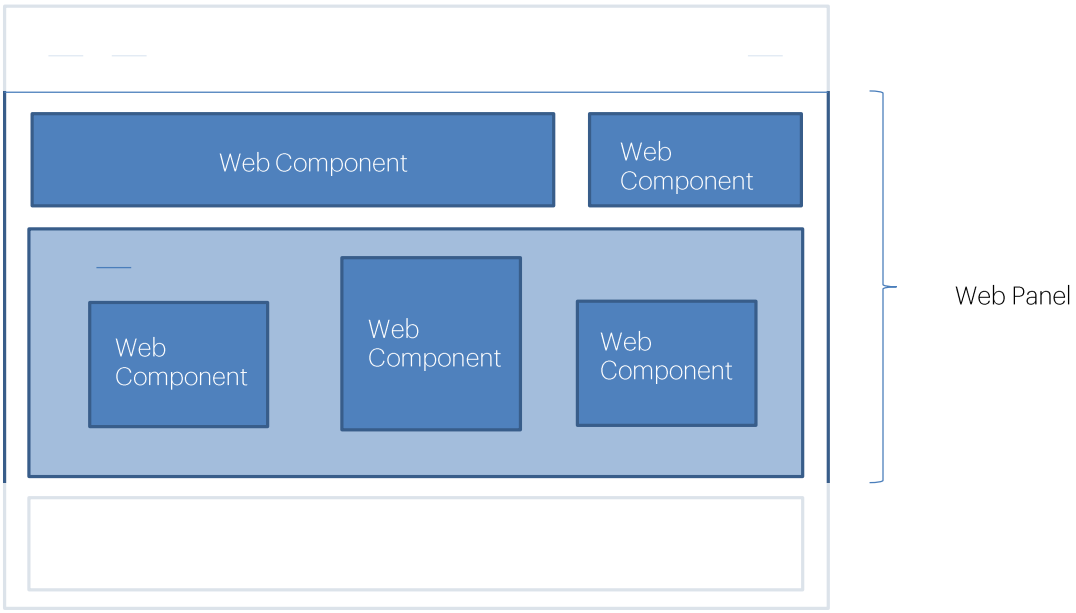
Screen composed by many screens

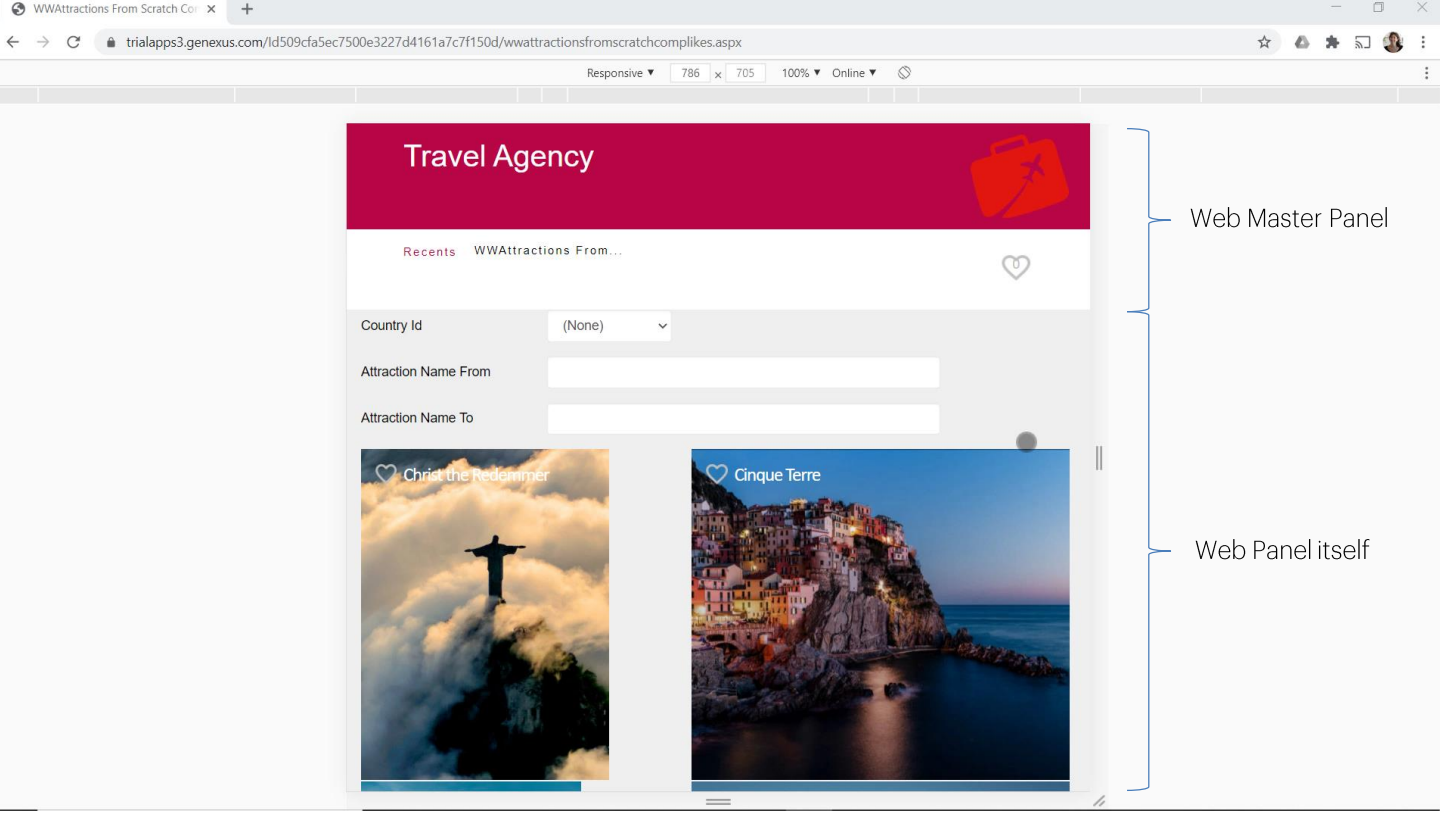


Como ya sabemos, a efectos de reutilizar, solemos *componentizar* lo más posible.

Así, si observamos cualquiera de nuestras pantallas en general corresponden a varios objetos interactuando y no a uno solo.

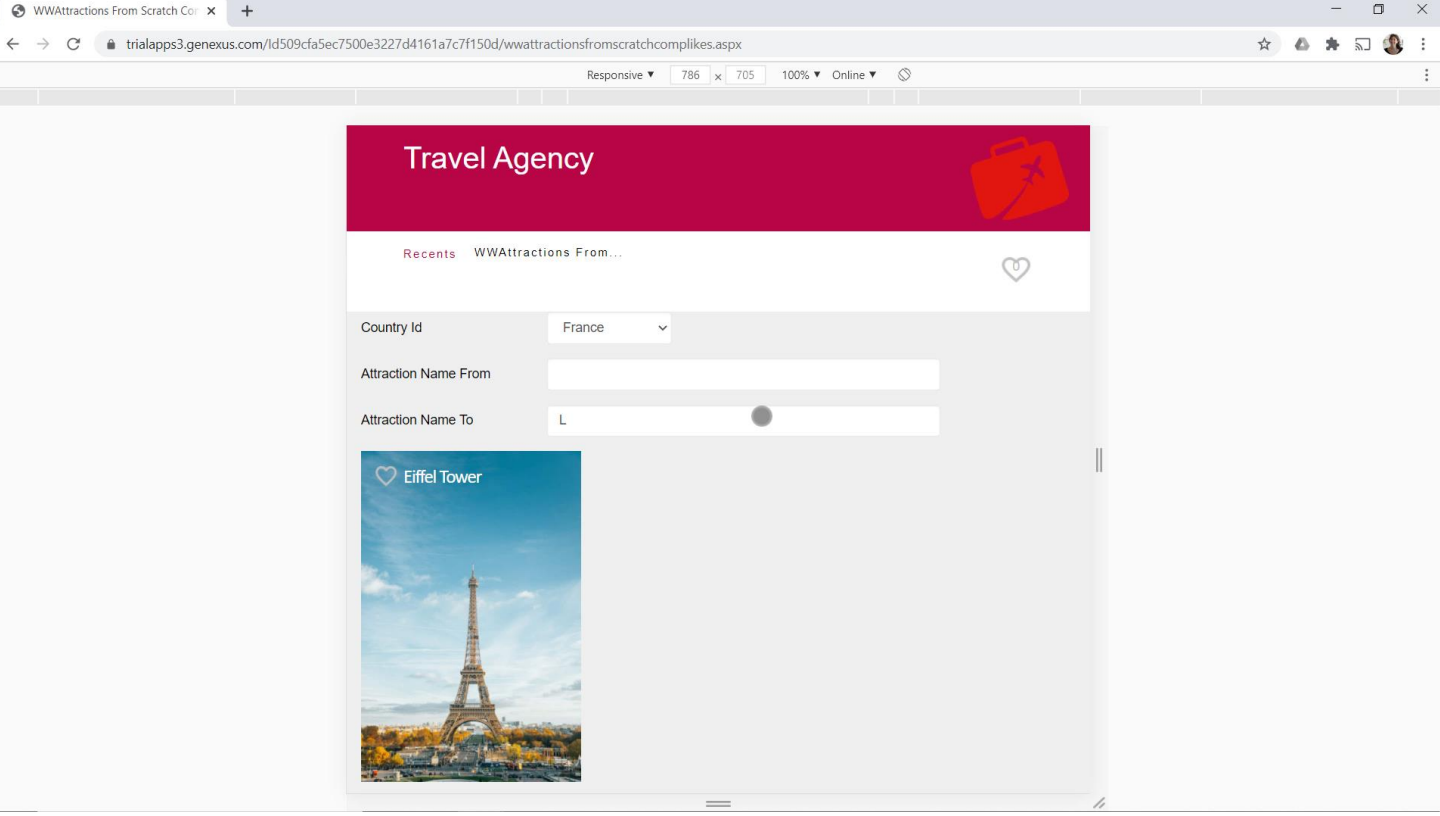
Screen composed by many screens



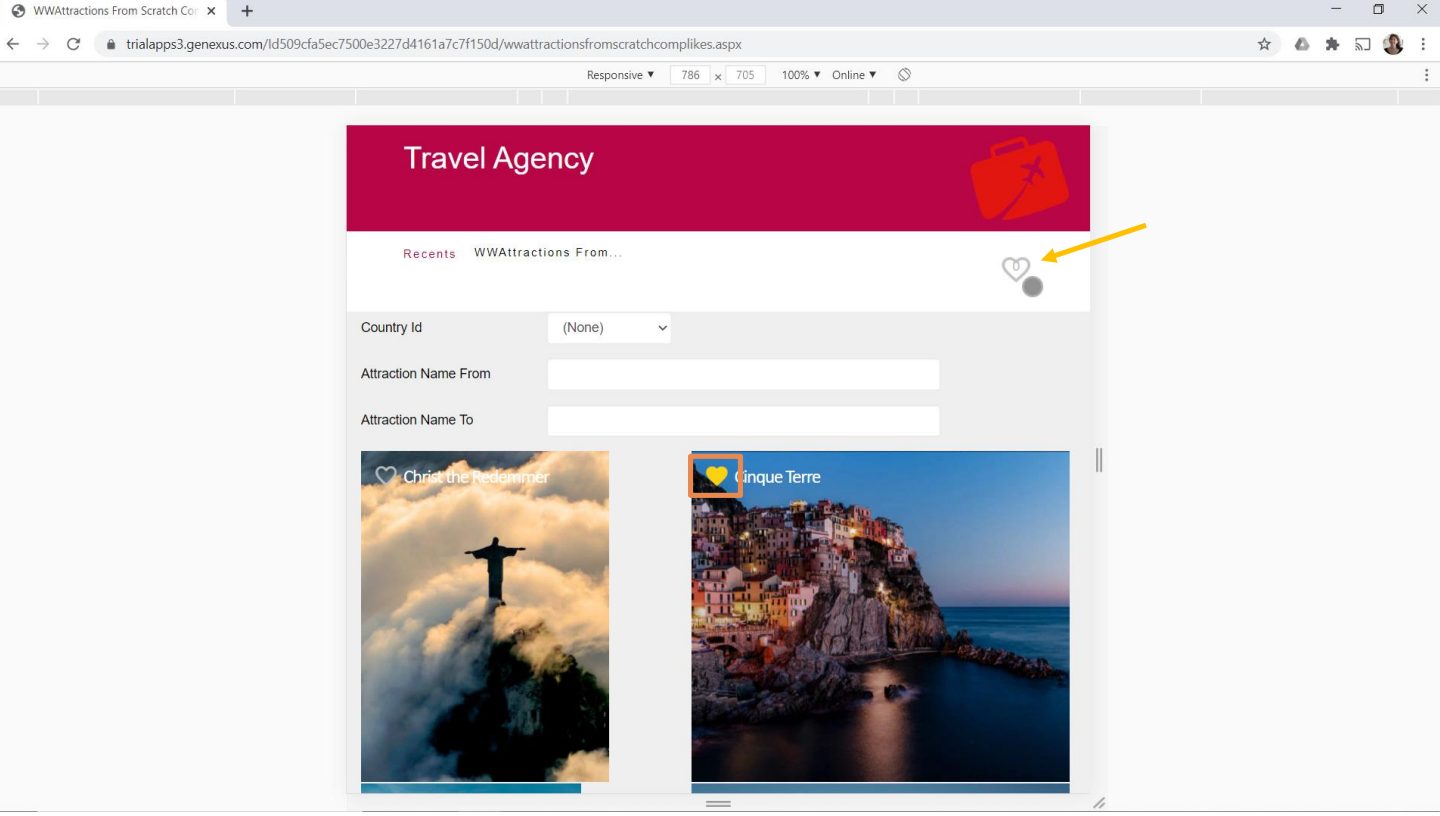


Por ejemplo, es el caso de un web panel como el que vimos en el curso inicial de GeneXus, donde se listaban las atracciones turísticas que una agencia de viajes ofrecía visitar. No nos ocuparemos aquí del diseño, que está apenas esbozado y le falta trabajo.

Estamos viendo una pantalla no demasiado compleja, que tendrá toda una parte que vendrá del panel maestro, y otra que es propia. Entre la propia, hay partes implementadas, a su vez, como componentes, aunque en ejecución esto sea inapreciable.



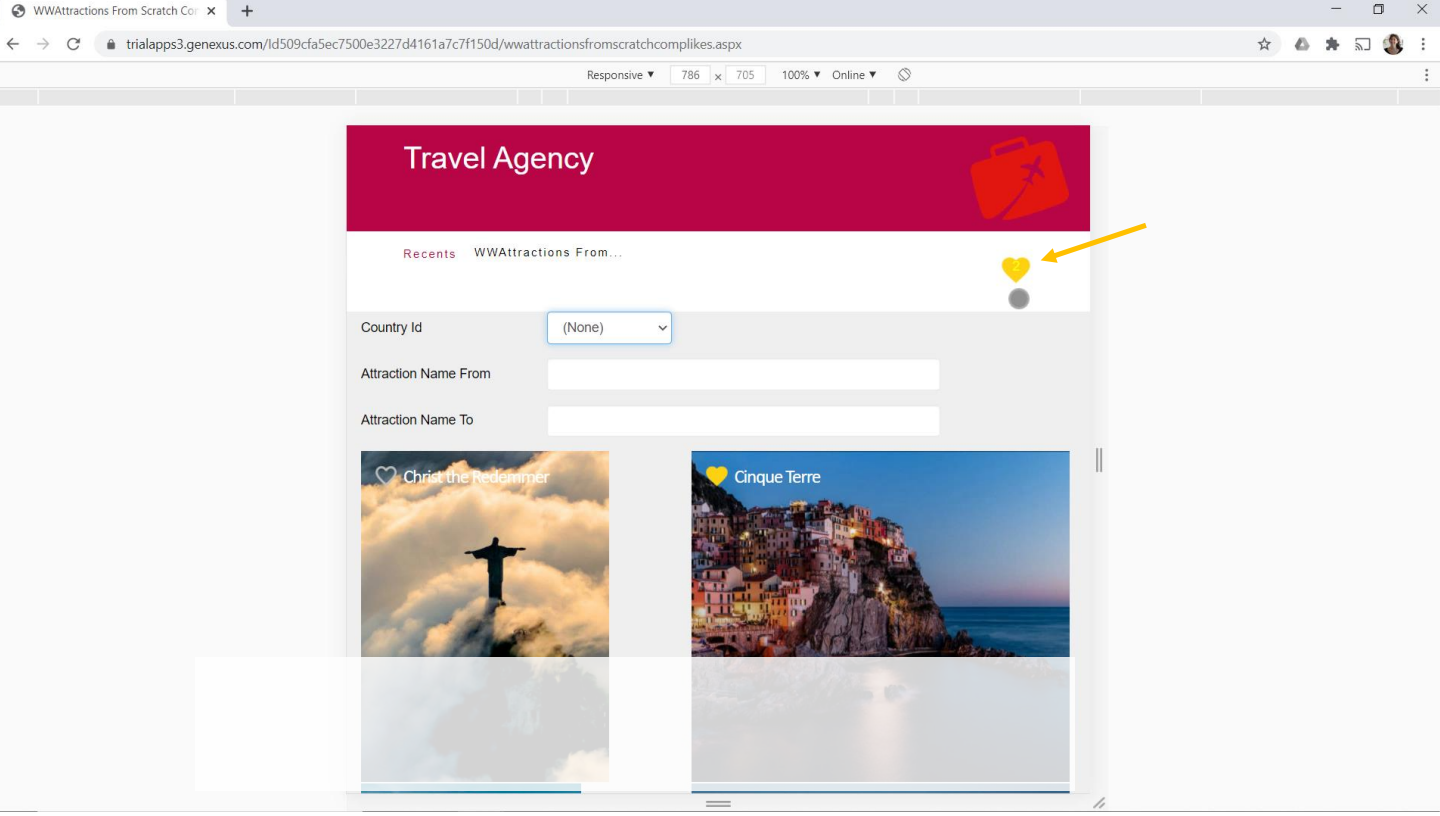
Podemos filtrar por país, y también por nombre de atracción.



Pero lo más interesante, que es lo que nos permitirá introducir el tema de este video, es que vemos que hemos agregado la posibilidad de *favoritear* atracciones turísticas.

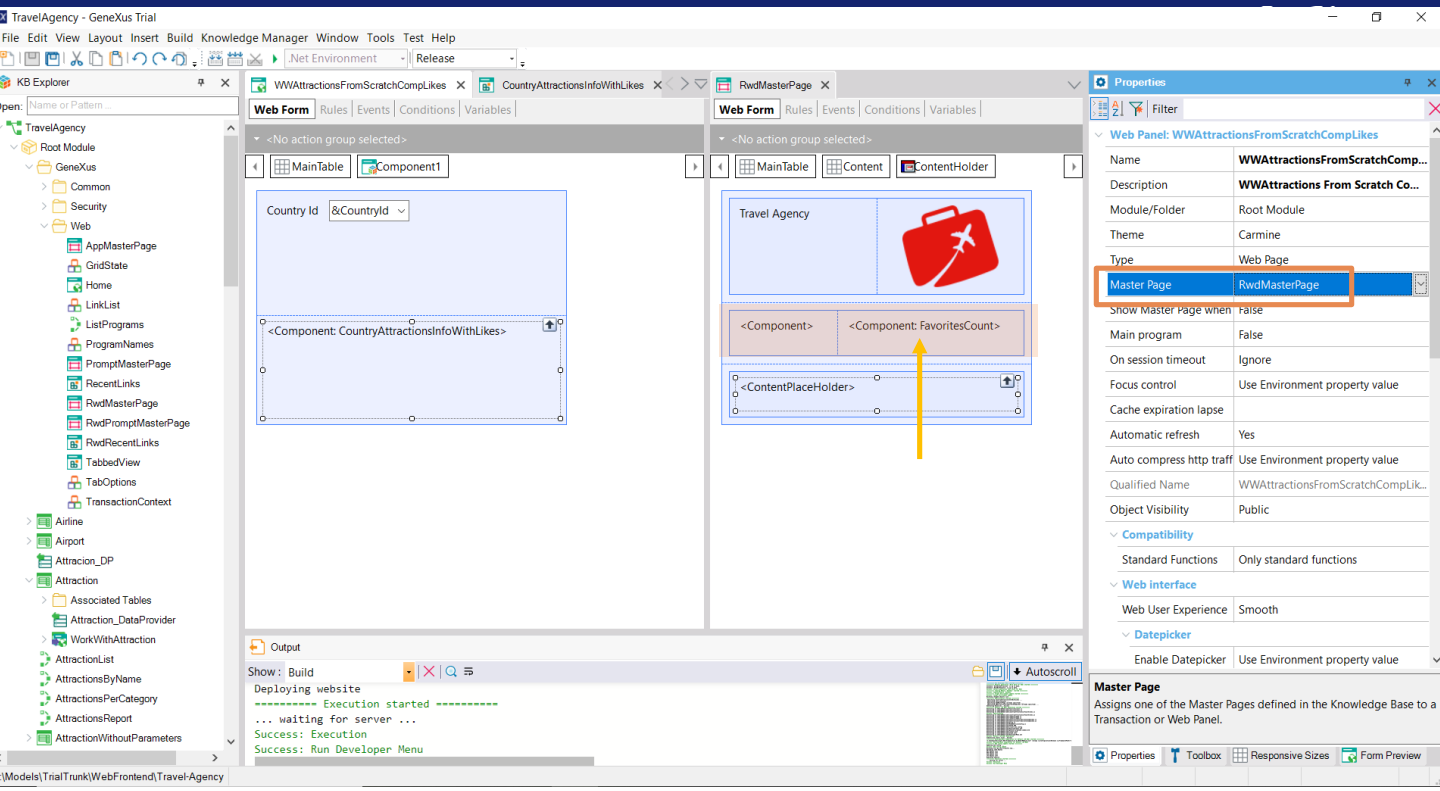
Por ejemplo, marco esta, y esta otra. Es decir, hemos marcado dos atracciones como favoritas.

Sin embargo, si vamos a observar aquí arriba, nos está indicando un cero, como si no tuviéramos favoritas.



Refresquemos la pantalla. Ahora sí vemos ese 2. Pretenderíamos, también, que si quito un favorito, este número se refresque, marcando 1.

La pregunta que nos va a guiar es cómo conseguimos que ese valor se actualice automáticamente conforme se *favoritea* o *des-favoritea* cada atracción.

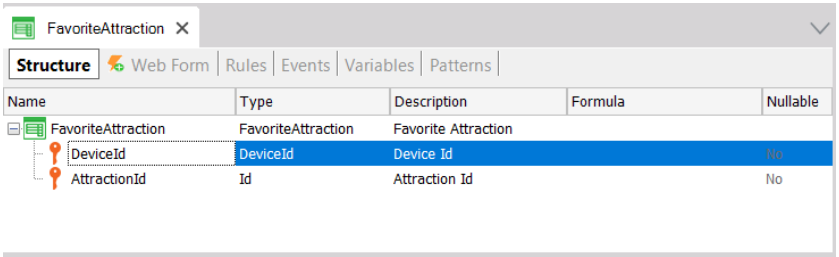


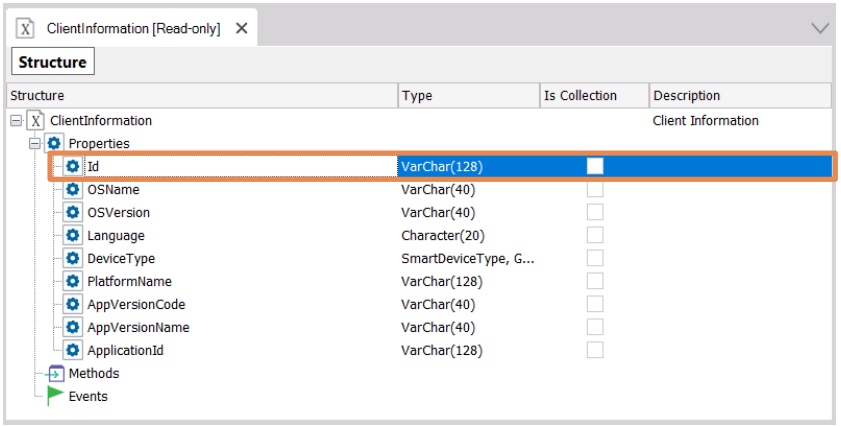
Vamos a GeneXus a ver este Web panel.

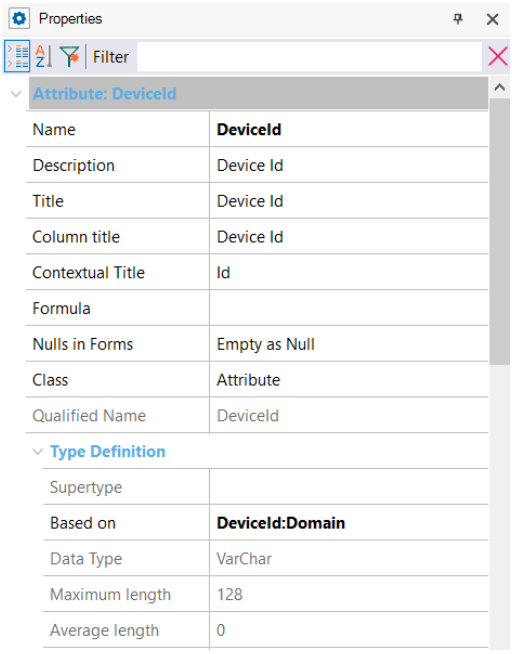
Vemos, primero que nada, que tiene un panel maestro asociado, que es el default. Por lo que, el contenido de su pantalla se cargará dentro del content place holder de la página maestra.

Y fuera de él estará todo esto otro.

En particular tenemos dos componentes, uno default, que es el que implementa los links recientes y el otro es uno que hemos agregado para mostrar la cantidad total de atracciones turísticas que el usuario ha *favoriteado*.





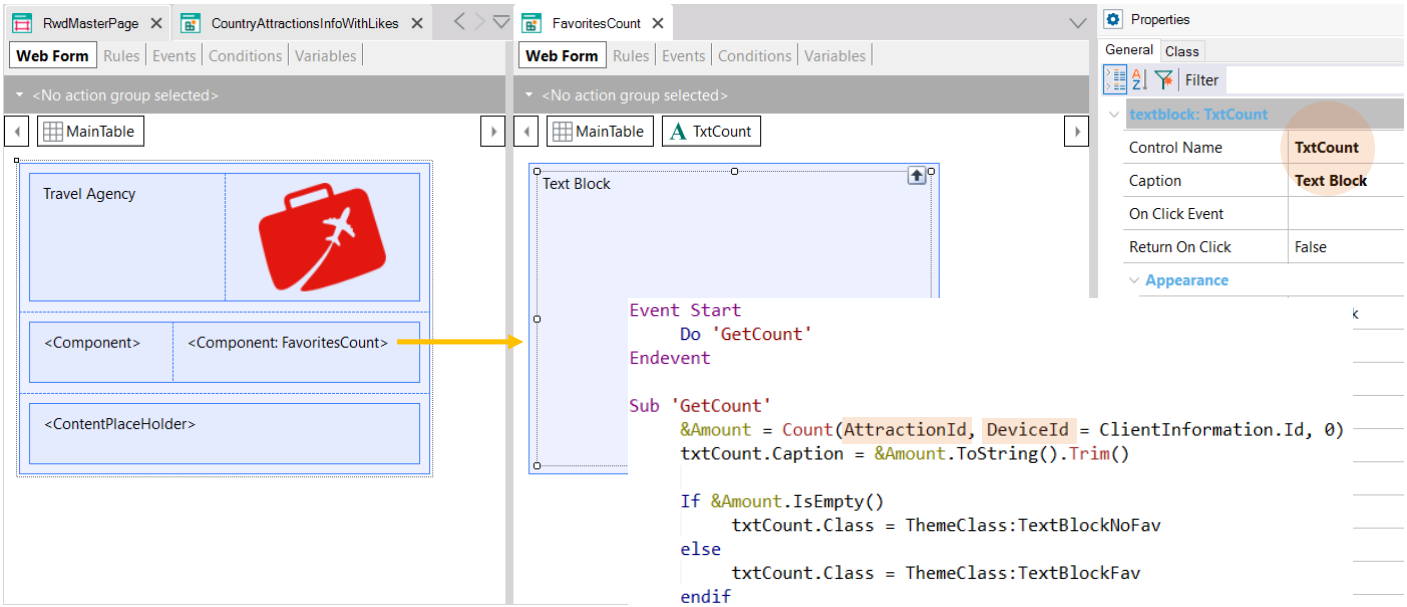


Para guardar las atracciones que un usuario *favoritea* deberíamos contar con una tabla de usuarios, para poder indicar atracciones favoritas por usuario. Si no queremos aún trabajar con usuarios (por ejemplo porque aún no le aplicamos GAM y estamos todavía resolviendo si vamos a tener una tabla propia de usuarios o no) entonces podemos provisoriamente mantener favoritos por instancia de browser.

Para ello creamos esta transacción con este atributo, al que le hemos especificado este nuevo dominio que también hemos creado nosotros, con el mismo tipo de datos que esta propiedad...

ClientInfomation es un external object del módulo GeneXus, cuya propiedad Id permite identificar al dispositivo cliente que está ejecutando, tanto sea en forma nativa como web.

En el caso de aplicaciones Web, la propiedad devuelve un identificador del usuario, que persiste entre todas las sesiones con el mismo navegador y para la misma aplicación. Entonces lo que hemos hecho es agregar una transacción con clave compuesta por el dispositivo y la atracción turística. En la tabla de esta transacción guardaremos los favoritos.

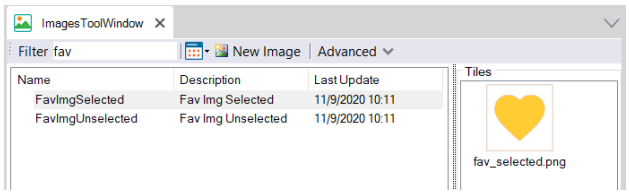


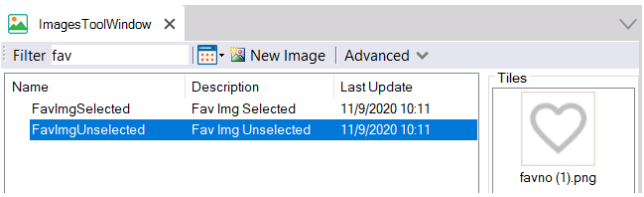
Si observamos el Web component que estamos cargando aquí, en el Master Panel web, vemos que solo contiene un text block, de este nombre, cuyo caption cargamos en ejecución, en principio solo en el evento Start, invocando a esta subrutina que lo primero que hace es calcular la cantidad de atracciones turísticas para este cliente, que se encuentran en la tabla que contiene a estos dos atributos (que es FavoriteAttraction).

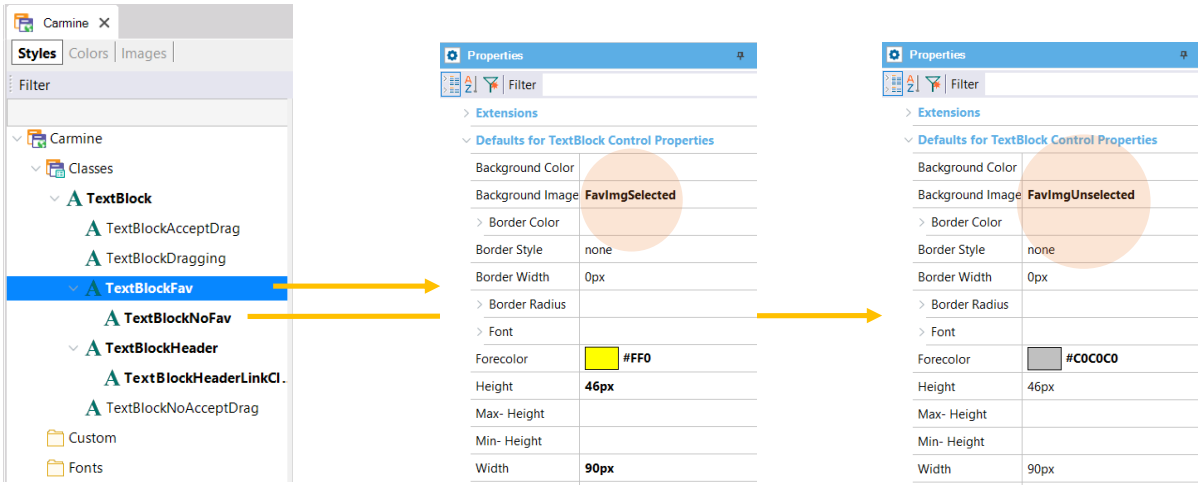
Aquí tenemos el valor que estamos buscando. Luego a ese numérico lo transformamos en string, para asignárselo al textblock que se mostrará.

Y para mostrarlo con la imagen de favorito y en el color y formato adecuado, hemos agregado dos clases al theme correspondiente –que en este caso es Carmine-.

Si la cantidad de atracciones es cero, al text block le asignamos la clase que diseña el favorito vacío y en caso contrario, la que diseña el favorito amarillo relleno.

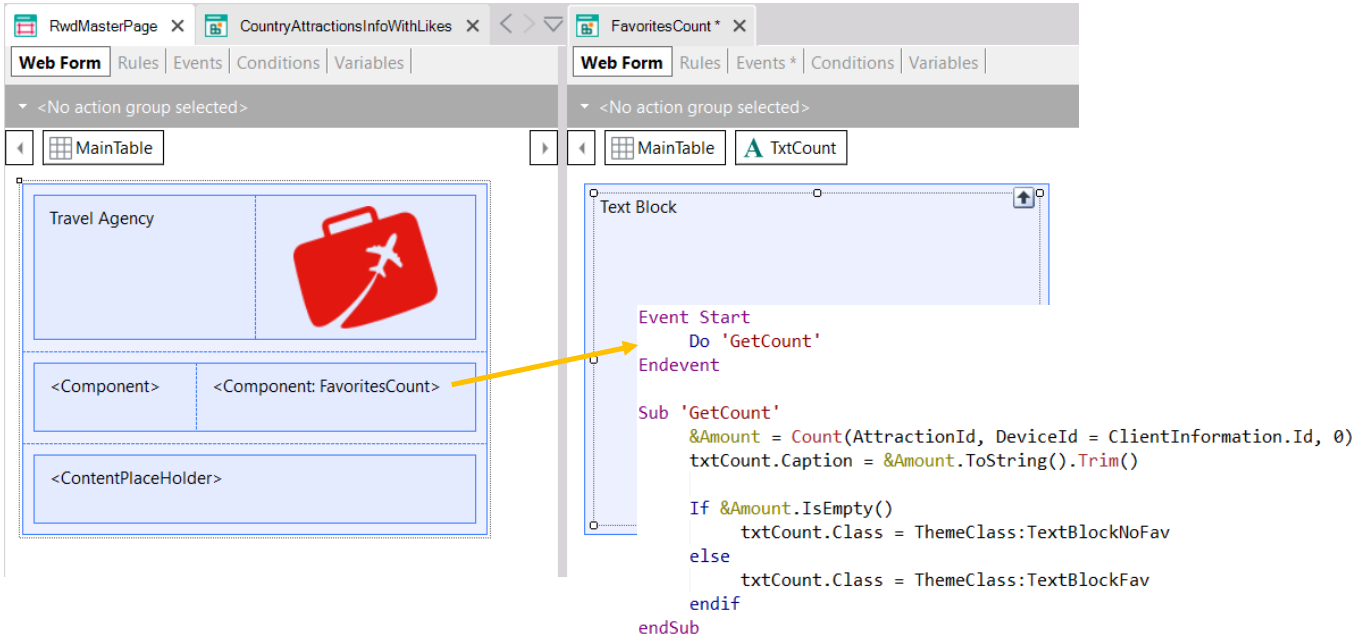






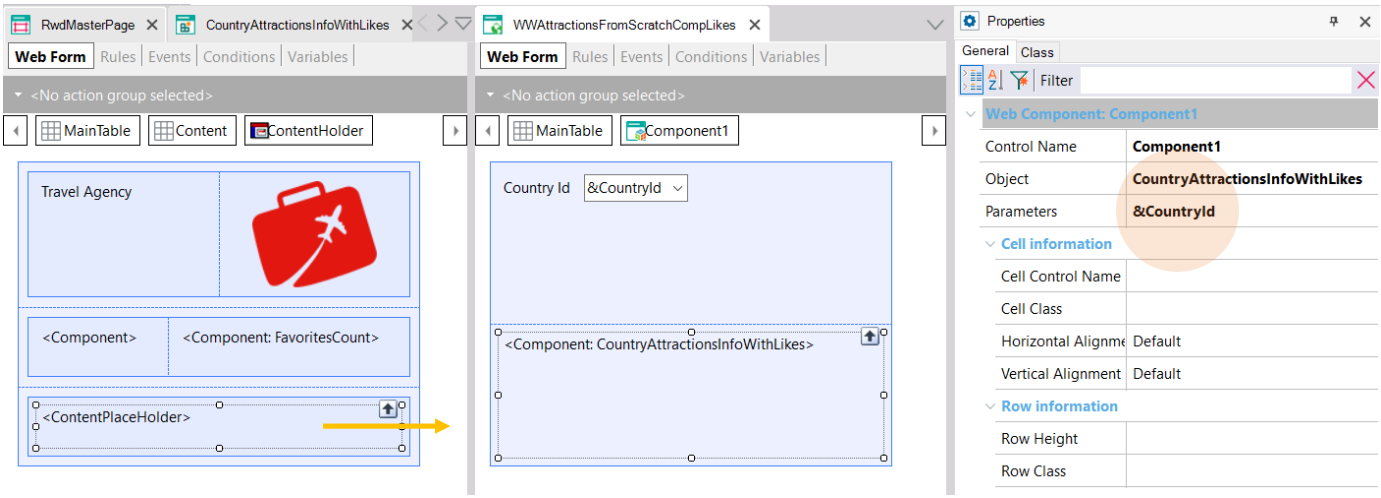
Para esto tuvimos que insertar previamente dos imágenes en la KB...

Si vemos las clases en el theme, aquí se indica en la propiedad Background Image el corazón amarillo lleno y para esta otra clase el gris vacío.



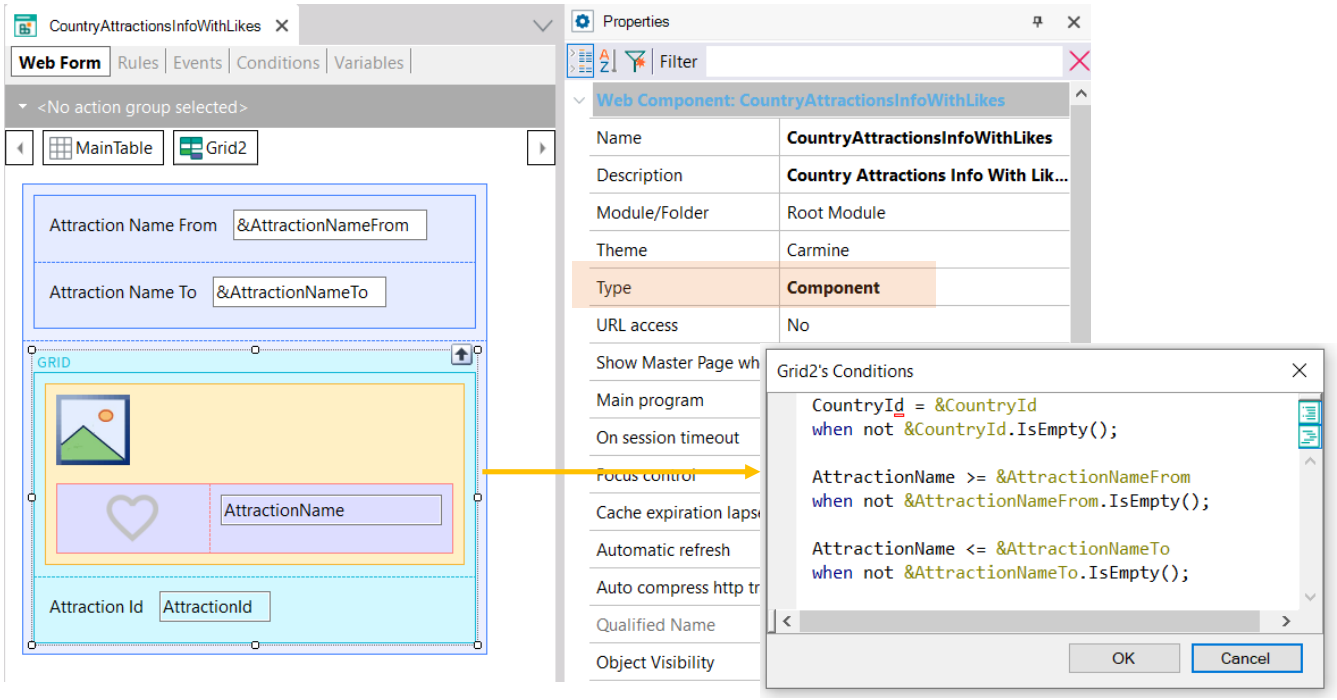
Así que este text block calcula la cantidad y la muestra con la imagen de fondo y los colores apropiados.

Este evento Start se va a disparar cuando se abra el panel maestro.

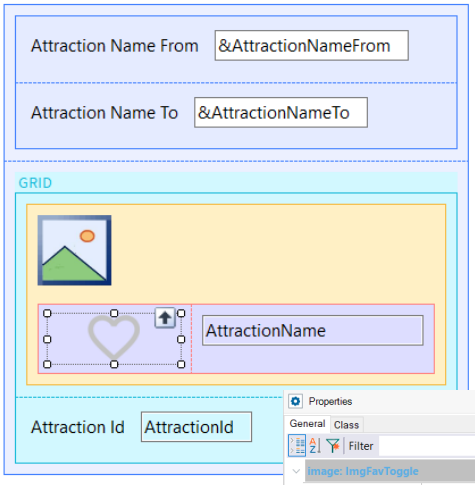


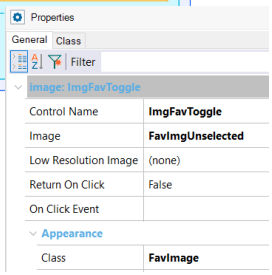

```
Event &CountryId.ControlValueChanged
    Component1.Object = CountryAttractionsInfoWithLikes.Create(&CountryId)
Endevent
```

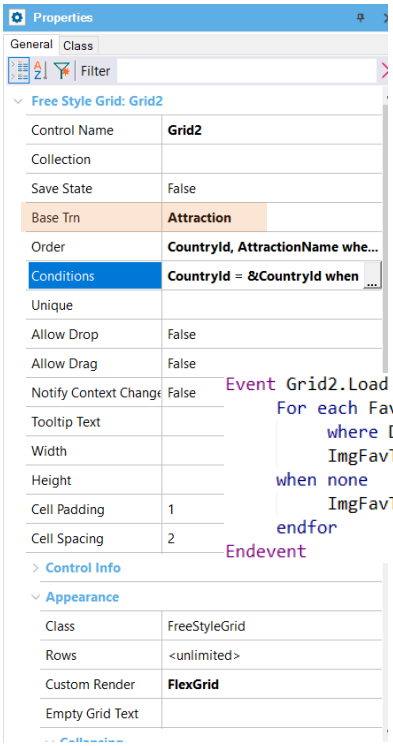
Por otro lado, si observamos lo que se cargará en el ContentPlaceHolder para nuestro Web panel... vemos que es una pantalla con este combo box para elegir un país, y debajo tenemos otro componente, al que se le envía el valor de ese país, toda vez que lo modifiquemos.



Y si observamos quién se está cargando en ese web component, es este otro objeto web, de tipo componente, que tiene un grid que filtra las atracciones turísticas que mostrará, de acuerdo al país recibido por parámetro y a los filtros de la propia pantalla.



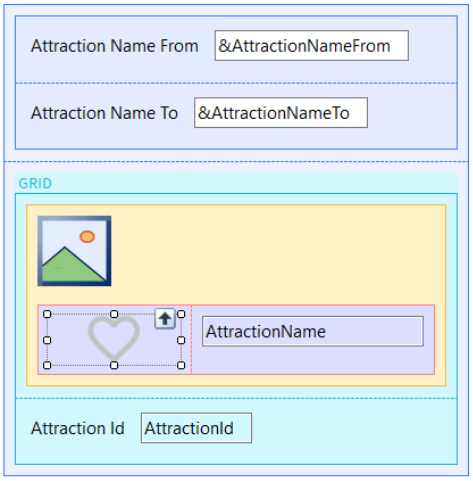





```
Event Grid2.Load
    For each FavoriteAttraction
        where DeviceId = ClientInformation.Id
        ImgFavToggle.FromImage(FavImgSelected)
    when none
        ImgFavToggle.FromImage(FavImgUnselected)
    endfor
Endevent
```

Hemos elegido un grid Flex, que contiene una tabla Canvas, para poder superponer la foto de la atracción turística junto con su nombre y una imagen para permitir al usuario *favoritear* o *des-favoritear* la atracción. Sobre el diseño de todo esto no nos detendremos aquí.

Solo observemos que en el evento Load del grid, que tiene como tabla base a Attraction, para cada atracción turística que va a ser cargada se ejecuta este for each, que busca en la tabla subordinada, FavoriteAttraction, si existe un registro para este dispositivo, el que está ejecutando. Si existe, eso significa que la atracción es favorita, por lo que carga la imagen con el corazón lleno; y en caso contrario, con el corazón vacío.




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

ToggleFavorite

Source | Layout | Rules | Conditions | Variables

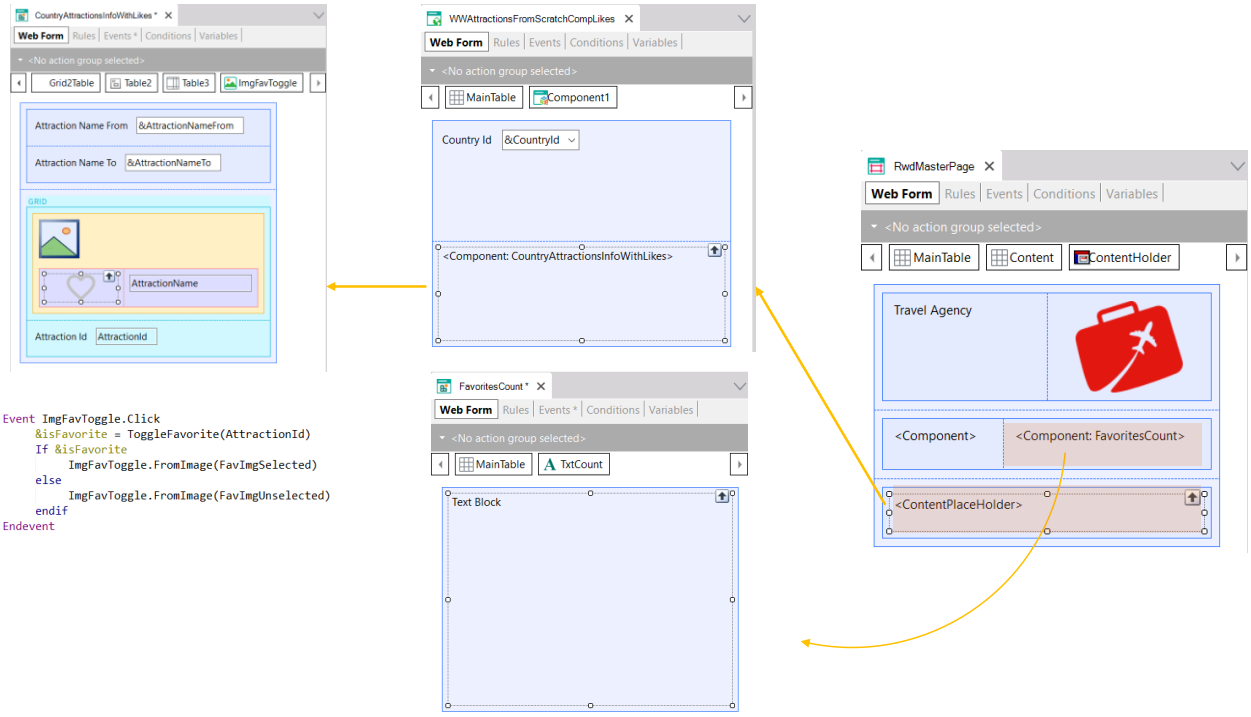
Subroutines

```
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
```

Pero de todo esto, lo que verdaderamente nos importa es otra cosa: la programación del evento click sobre esta imagen.

Una vez cargado el grid de atracciones, cuando el usuario hace clic sobre el favorito de una atracción, llamamos a un procedimiento que se fija si la atracción ya estaba *favoriteada*, en cuyo caso la elimina de la tabla e indica la operación para hacer el switch de la imagen por la vacía. Y si es al revés, hace lo contrario, inserta.

Ahora bien, al hacer esto, el total *favoriteado* debería actualizarse, pero no lo está haciendo.

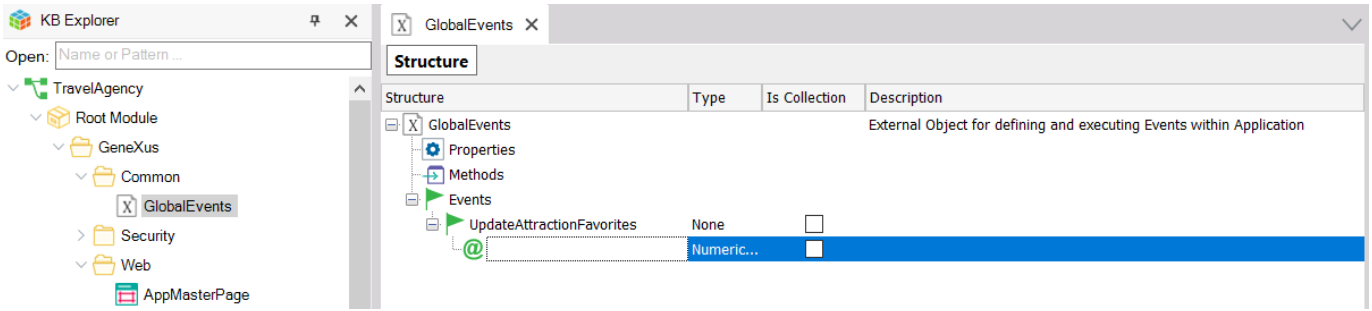



```
Event ImgFavToggle.Click
    &isFavorite = ToggleFavorite(AttractionId)
    If &isFavorite
        ImgFavToggle.FromImage(FavImgSelected)
    else
        ImgFavToggle.FromImage(FavImgUnselected)
    endif
Endevent
```

Recapitulando... este evento se dispara en un web component que está dentro de este web panel, que a su vez se ejecuta dentro del Web Master Panel este, que es donde se crea, también, el componente que queremos se refresque.

Dicho de otro modo: queremos que un evento de usuario de un componente cargado en este panel, permita realizar una acción sobre un componente con el que no guarda otra relación que el encontrarse indirectamente reunidos en la misma pantalla.

Global Events: default external object



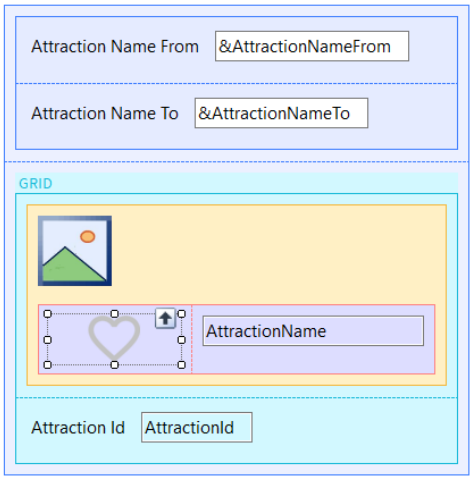
Para permitirles la interacción toda KB vendrá con este objeto predefinido, Global Events, para que podamos modificarlo.

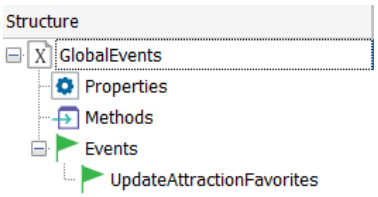
Podríamos salvarlo con otro nombre y crear tantos objetos particulares como queramos.

En nuestro caso vamos a modificar el predefinido, que viene vacío, agregándole un evento al que llamaremos así.

Los eventos que se vayan agregando podrán manejar parámetros, si se precisan para la comunicación entre componentes.

En nuestro caso solo necesitamos que uno se entere de que otro lo disparó. No necesitamos parámetros.



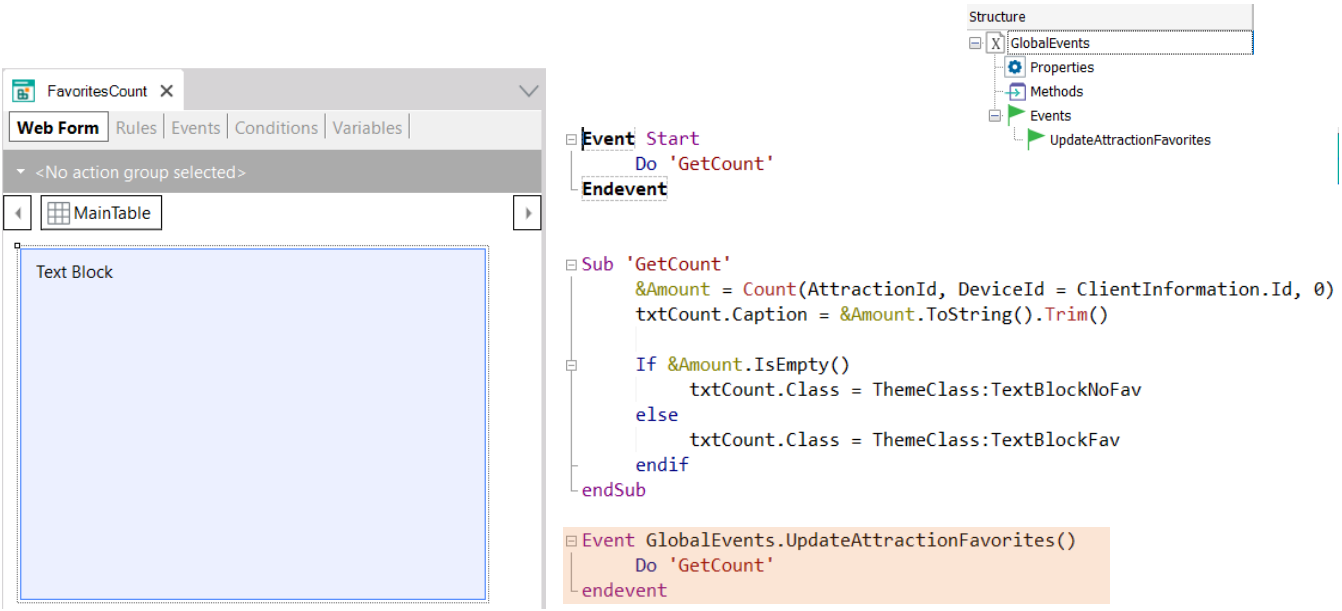



```
Event ImgFavToggle.Click

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

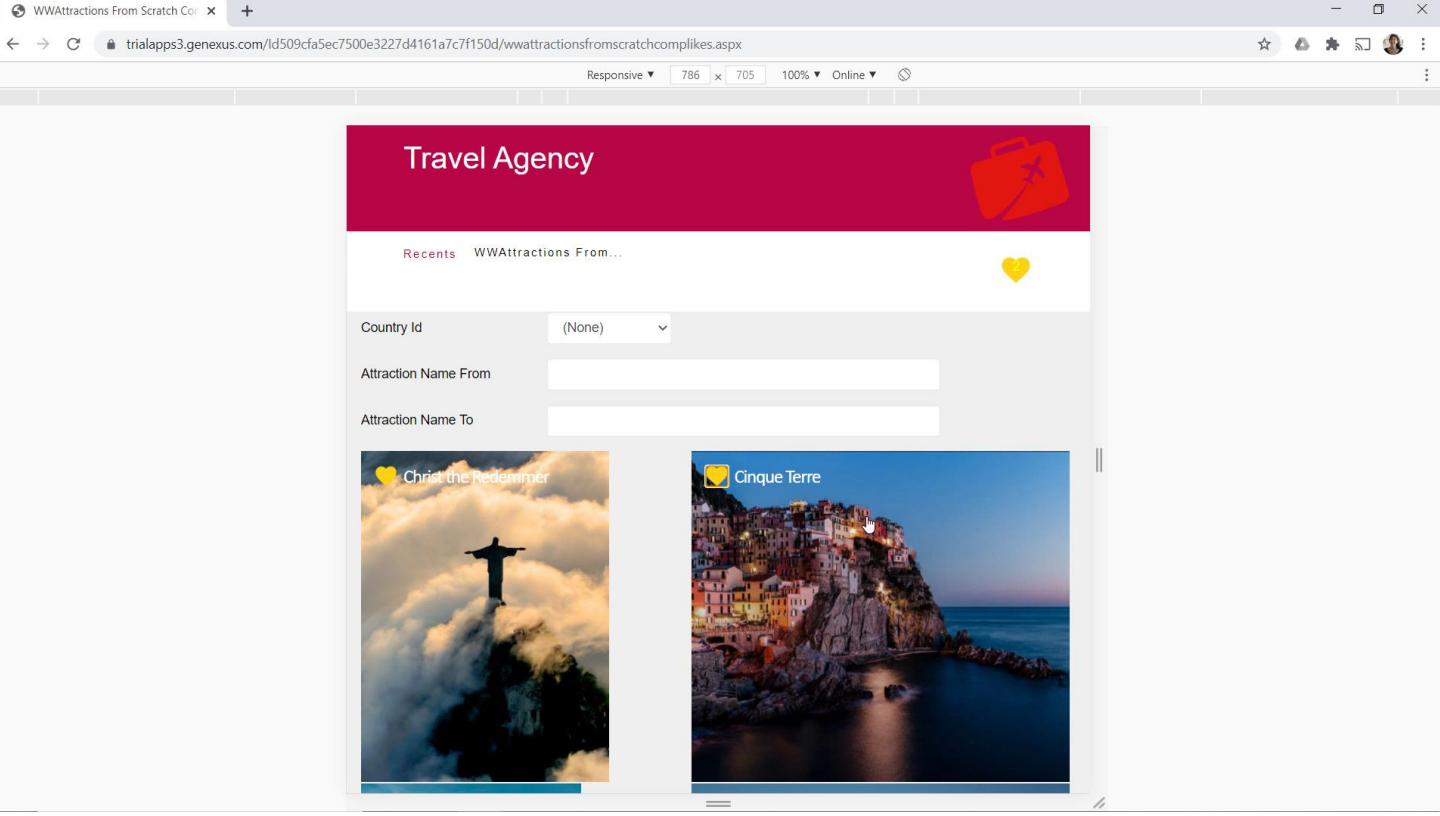
Teniendo esta forma global de comunicación, pongámosla a funcionar.

Vamos al panel que deberá disparar el evento. El disparo será cada vez que se haga clic sobre la imagen. En ese momento invocamos al objeto externo GlobalEvents, evento: el único que hay al momento. Con eso se está reportando el disparo del evento.

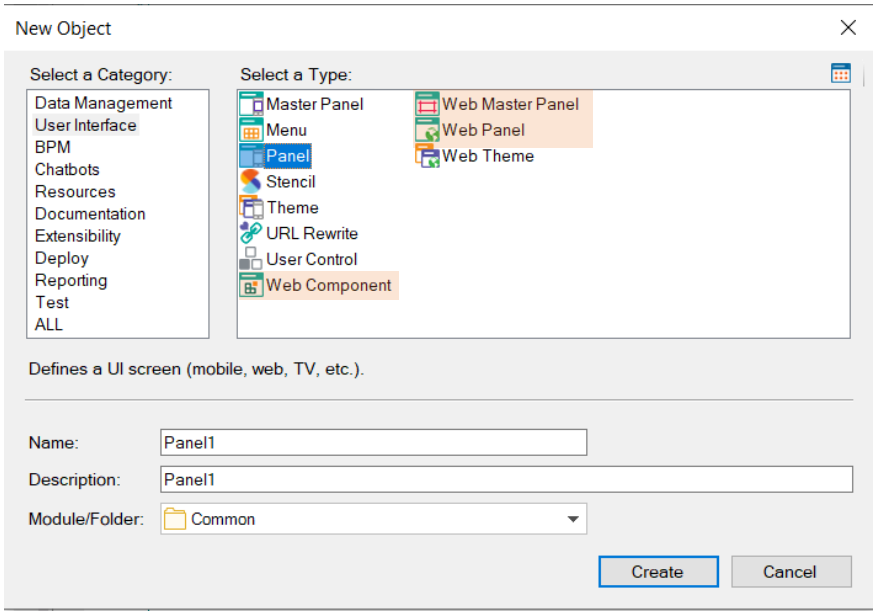


Luego, lo que tenemos que hacer es que el componente que cuenta las atracciones para mostrarlas se suscriba a este evento, para poder escucharlo toda vez que se produzca en la pantalla en la que quien escucha se encuentra.

Para ello simplemente declaramos escuchar el evento. Y allí programamos lo que queremos que se ejecute cuando el evento ocurre. En este caso es volver a calcular la cantidad de atracciones.



Ejecutemos para probar. Des-seleccionemos y vemos que se está refrescando. Ahora marcamos, marcamos... La comunicación está efectuándose con éxito.



Si bien aquí vimos un ejemplo de comunicación entre paneles web con components, lo mismo vale para comunicar paneles multi-experience (por ejemplo para aplicaciones nativas) con componentes.

Para aprender más sobre este tema, puede dirigirse a nuestro wiki.

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