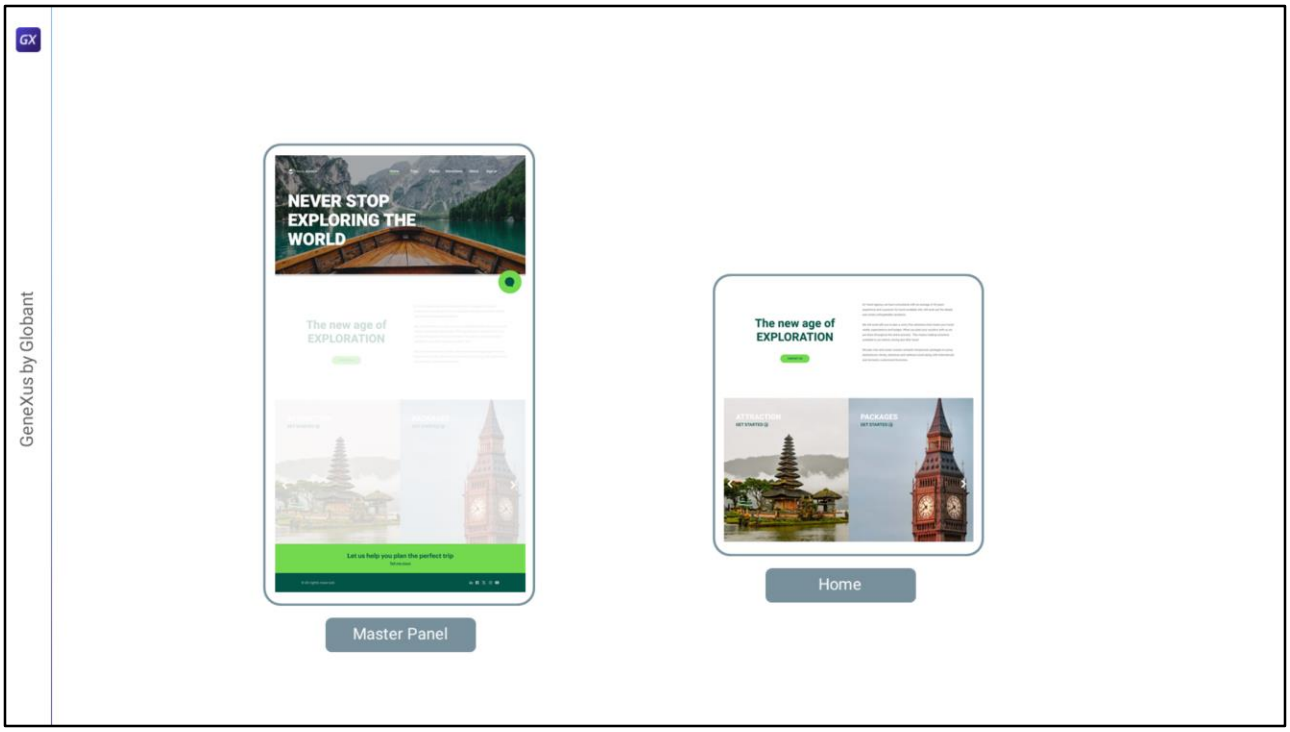
GeneXus by Globant

Master Panel:
Header update and Navigations

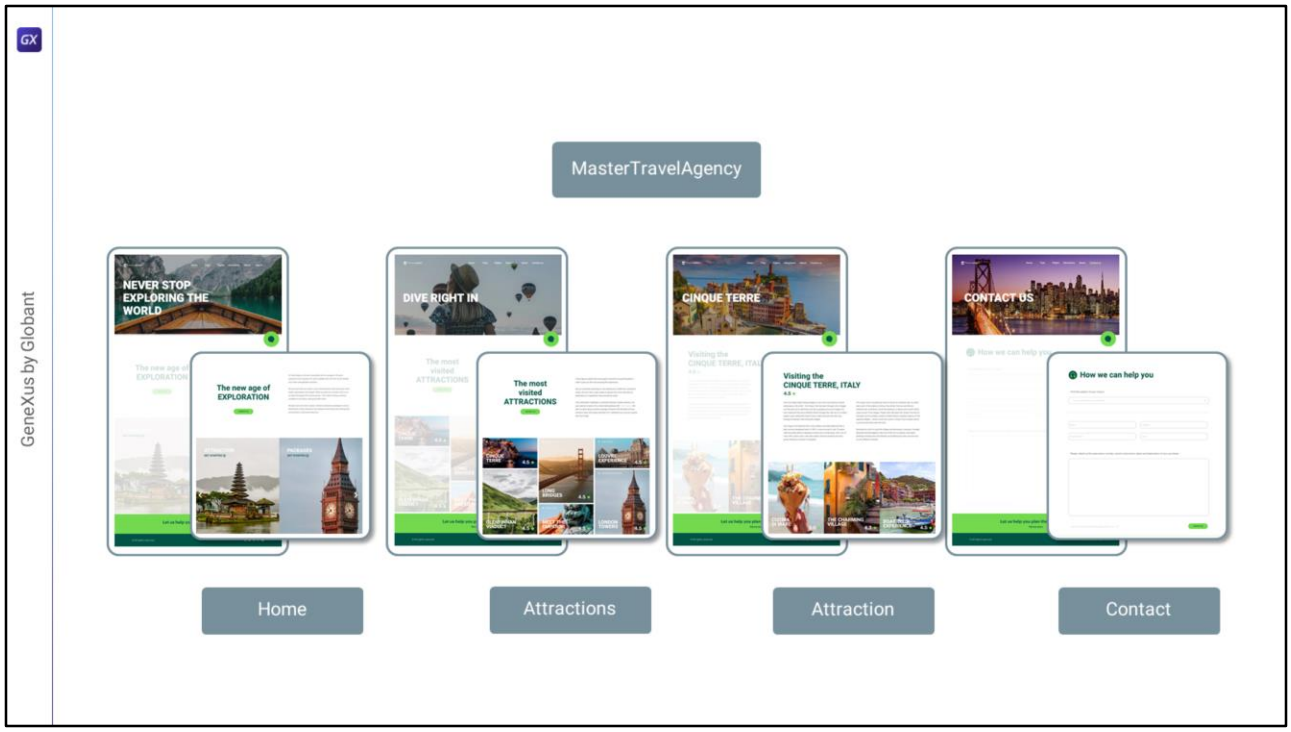
Cecilia Fernández



Bueno, en este video vamos a completar la implementación del Header.

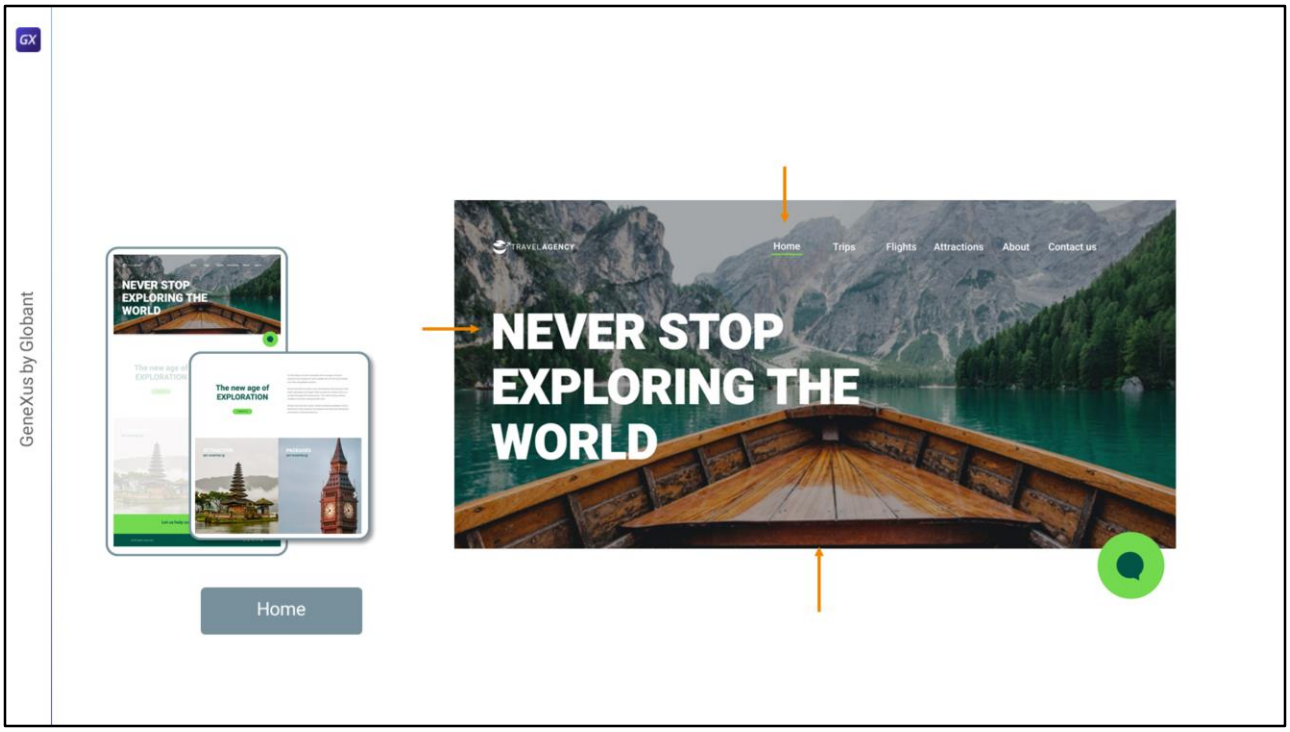
Como sabemos, el Master Panel no es un objeto ejecutable en forma independiente. El Master Panel se ejecuta toda vez que un panel que lo tiene como Master Panel se ejecuta.

Como ya sabemos, el layout del panel se renderizará dentro del control ContentPlaceHolder del Master Panel.

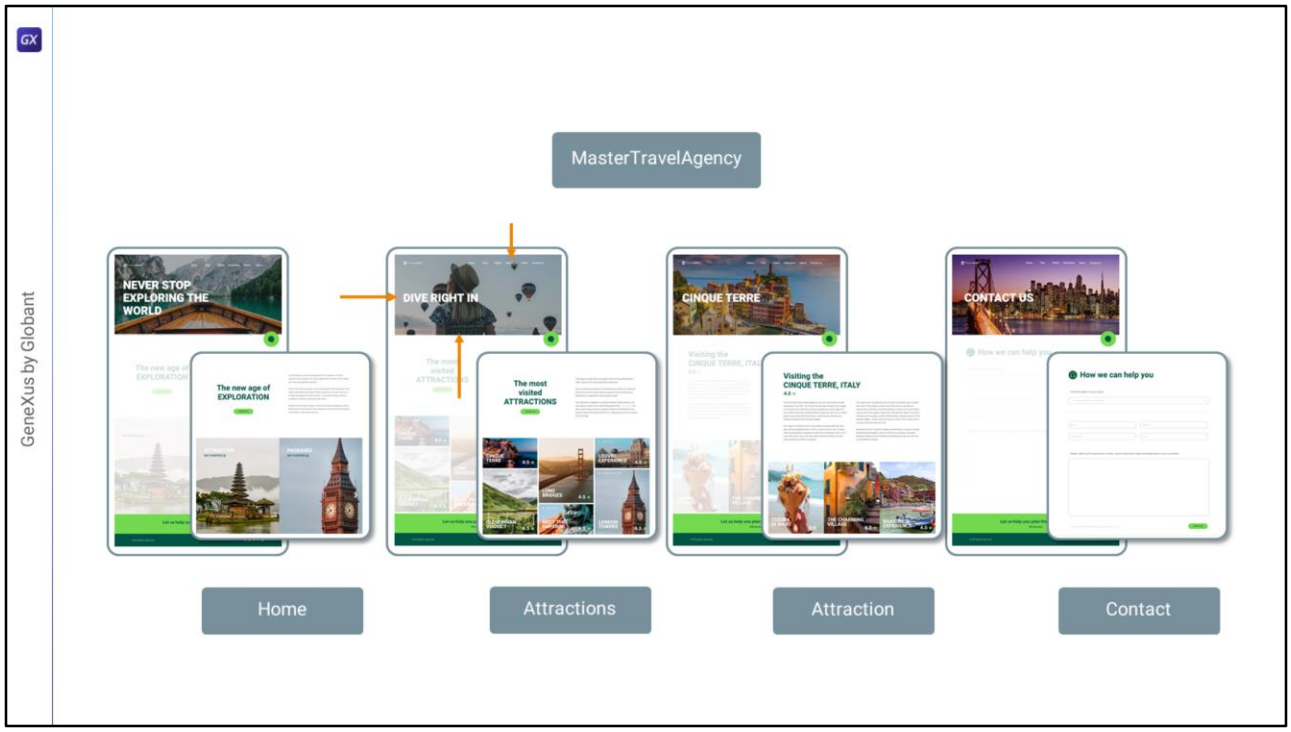


En nuestro caso, estos cuatro paneles serán los ejecutables. Pero cada vez que uno de ellos se invoque, ejecutará al Master Panel.

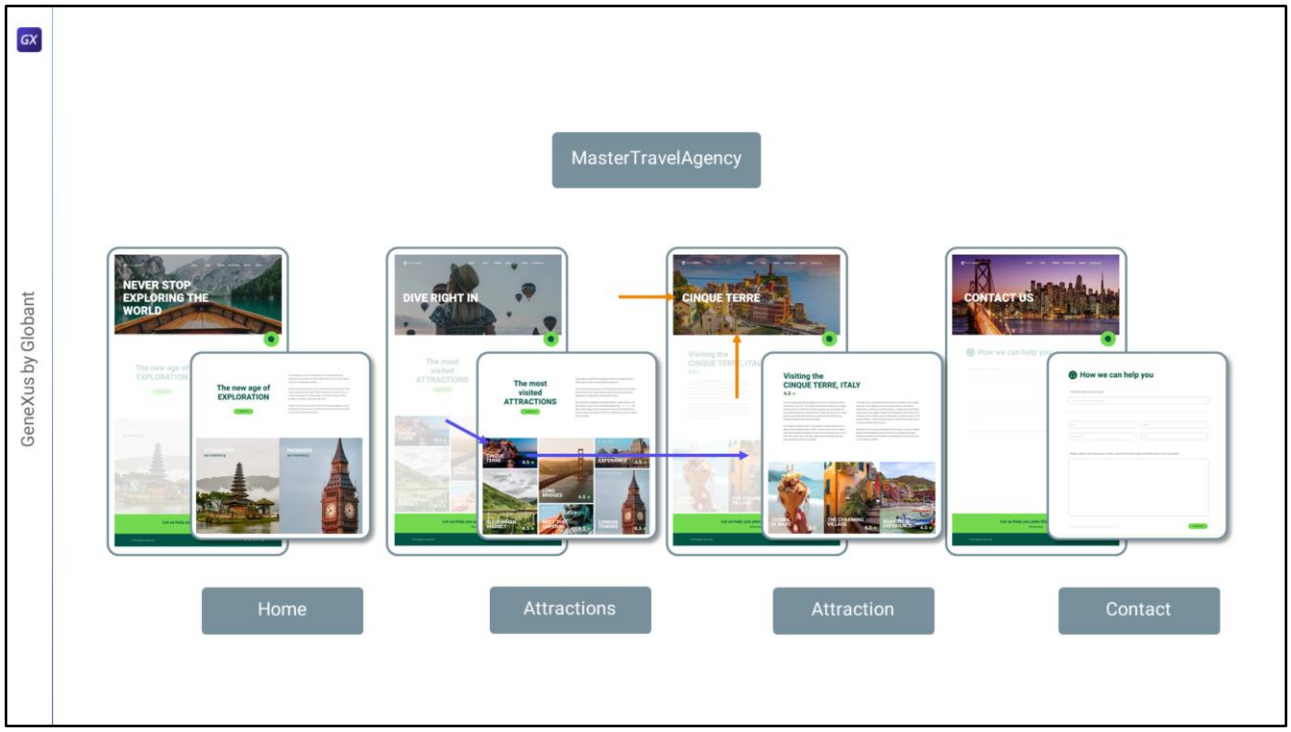
En nuestro caso el Master Panel deberá saber quién lo mandó a ejecutar porque ese “quién” determina tres cosas: la imagen de fondo del Header, el título sobre la imagen, y la opción del menú que debe quedar como la seleccionada.



Para el Home, deberá verse esta imagen, este título y el botón que deberá quedar como seleccionado (con el indicador verde debajo) es el Home.

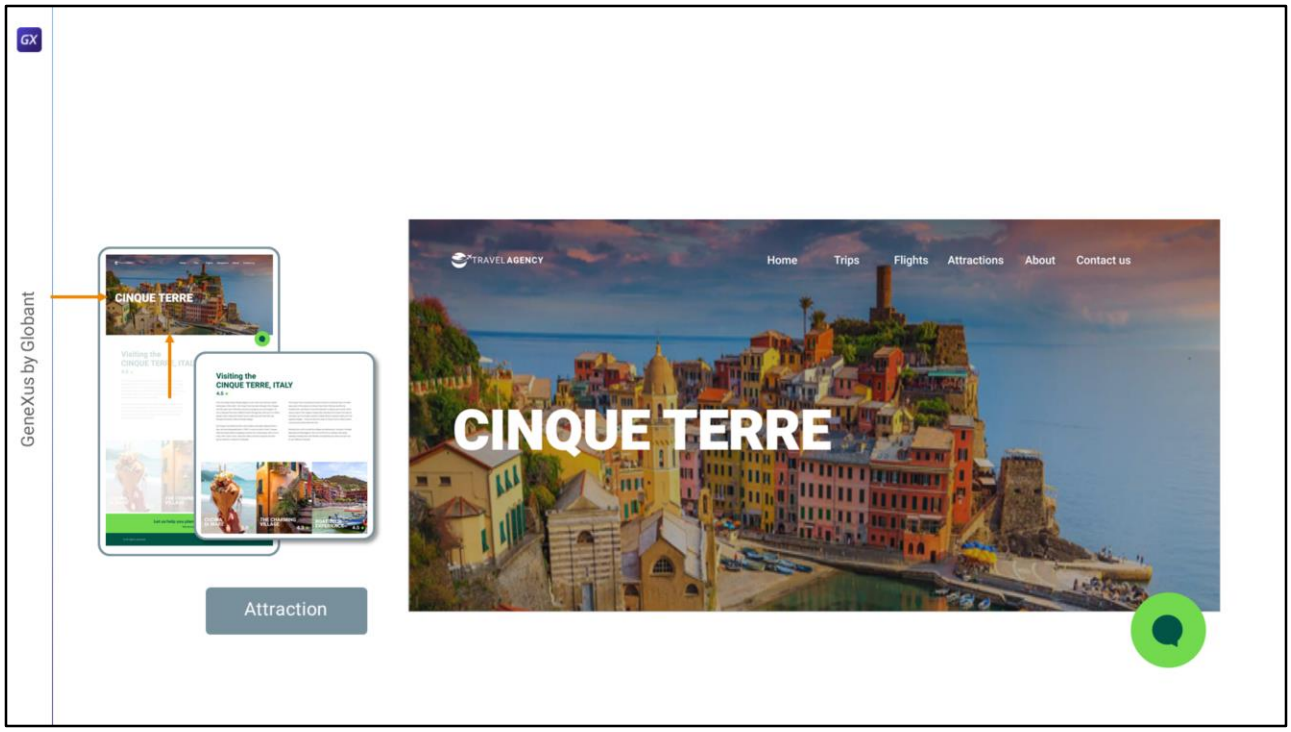


Para Attractions, estas deberán ser la imagen y el título y el botón seleccionado deberá ser Attractions.

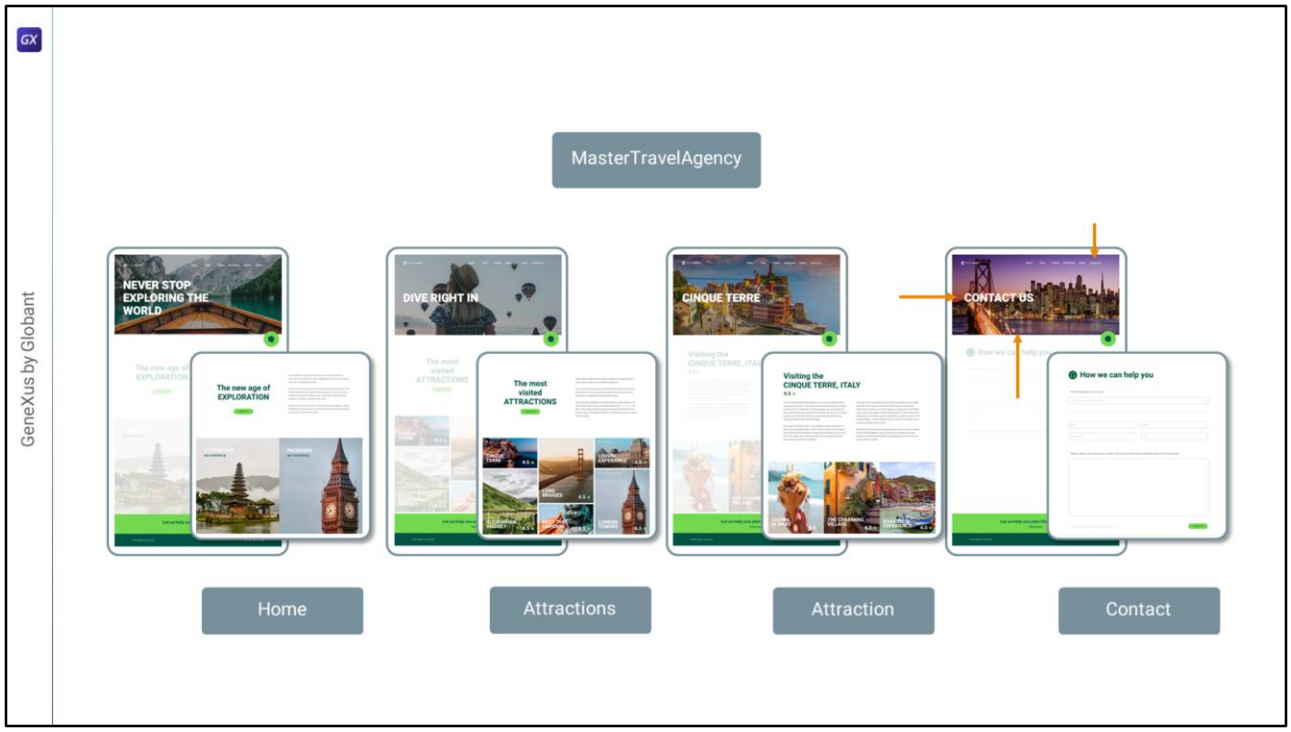


Para Attraction pasa algo diferente, porque no es una opción del menú. A esta página se accederá desde esta otra, eligiendo alguna de las atracciones turísticas mostradas en este carrusel. Entonces este panel recibirá el identificador de atracción turística por parámetro, y deberá acceder a la base de datos para cargar aquí la información de esa atracción turística.

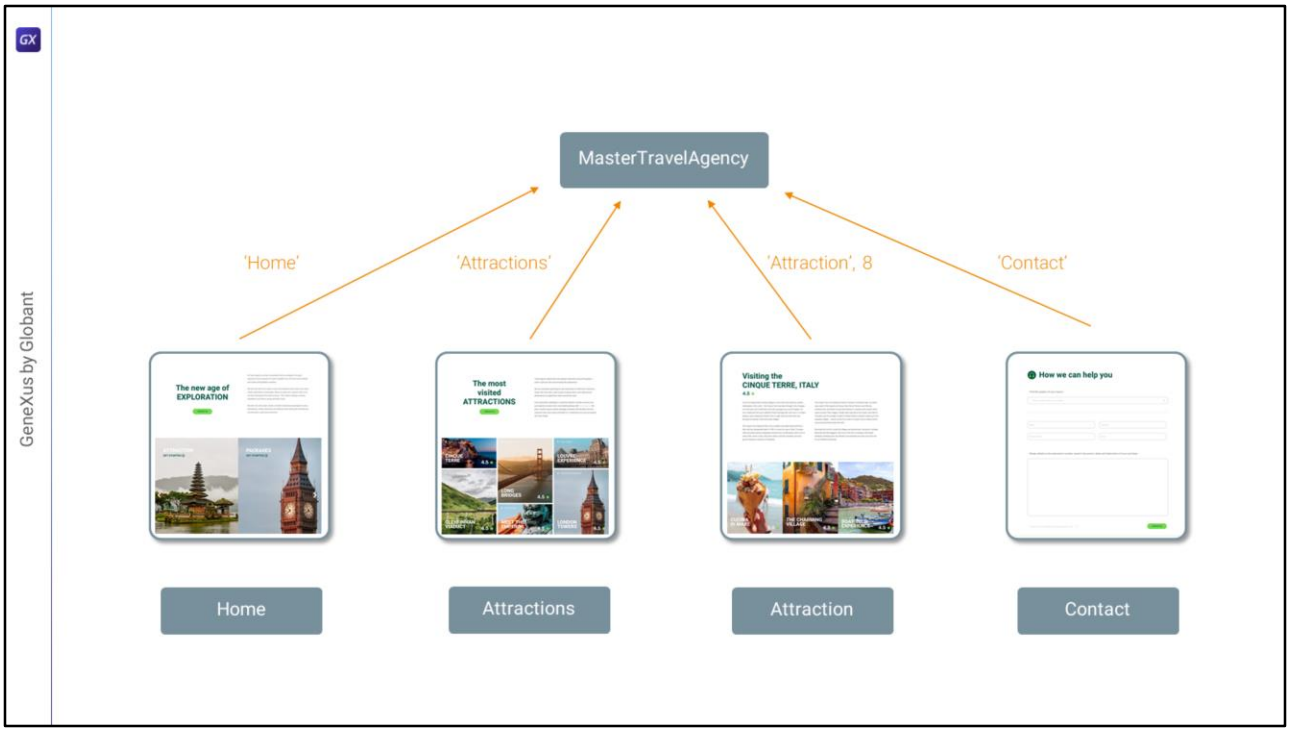
Pero además, a nivel del Master Panel, también deberá saber quién es esa atracción turística, porque también tendrá que ir a la base de datos para traer su foto, que es la que mostrará como Hero en el Header, y su nombre para mostrar como título…



…y en el caso del menú, en realidad deberán aparecer todas las opciones desmarcadas.



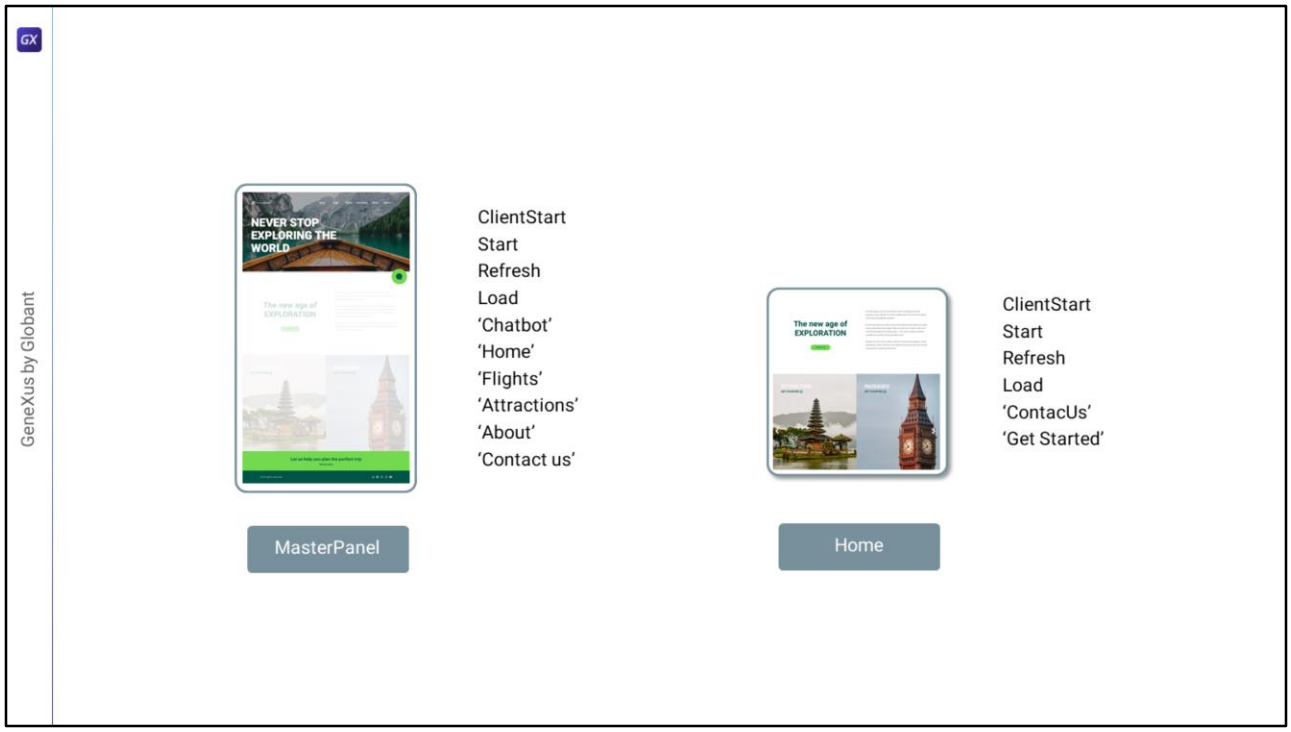
El panel del contacto es igual que los dos primeros, con imagen, texto y botón seleccionado fijos.



En definitiva, el panel que se esté ejecutando en cada oportunidad debe identificarse ante el Master Panel, para que éste pueda tomar las acciones necesarias sobre los 3 elementos del Header que acabamos de analizar: imagen de fondo, título y botón seleccionado.

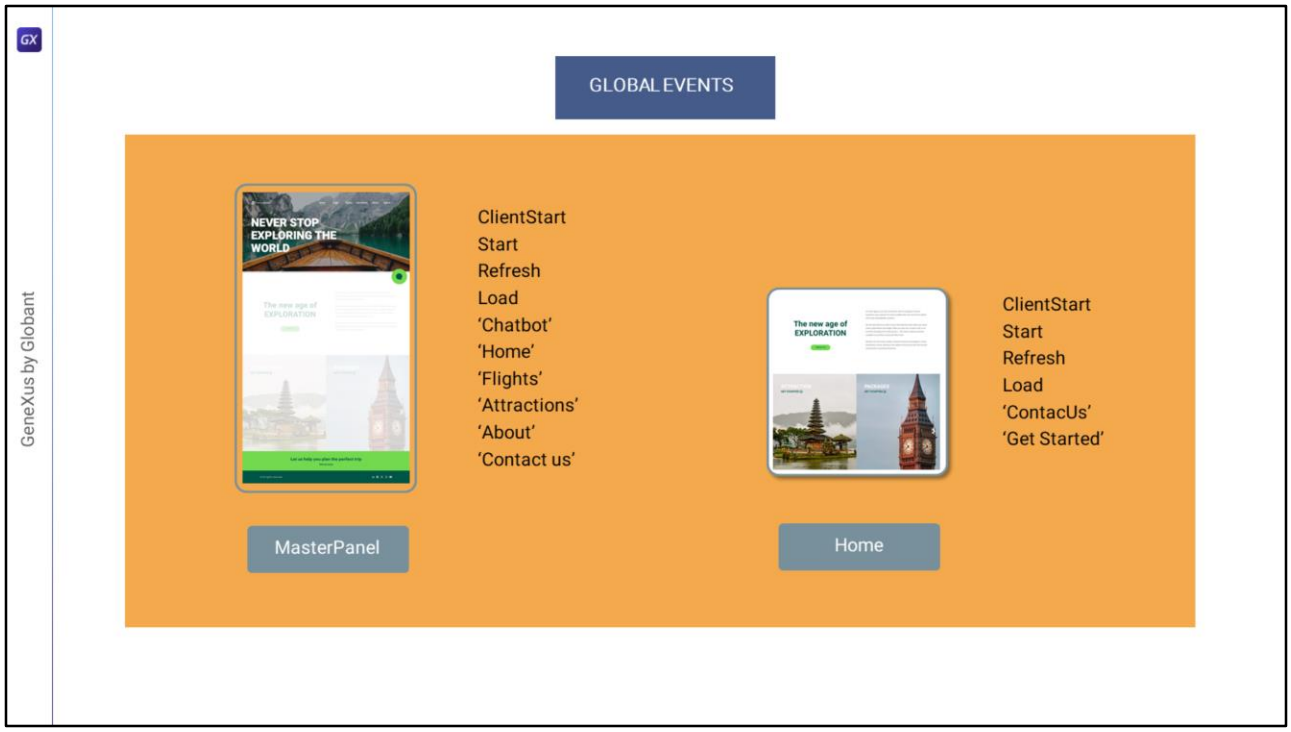
Podríamos pensar que esto se resuelve con pasaje de parámetros, es decir, cada panel le envía por parámetro al Master Panel una identificación de quién es. Pero el Master Panel no admite pasaje de parámetros.

¿Entonces? ¿Cómo podemos hacer para producir esta comunicación entre objetos que no se pueden pasar parámetros? Una buena opción es utilizar los eventos globales.

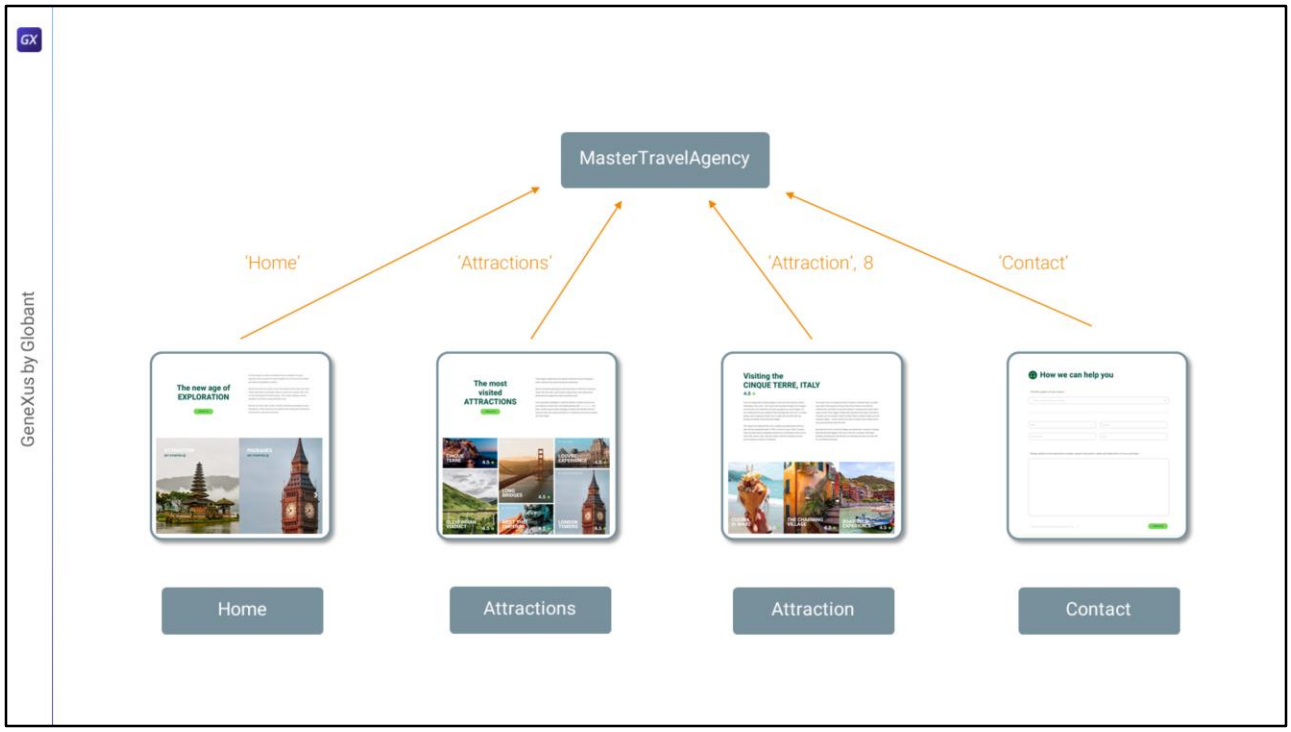


Los eventos con los que estamos familiarizados hasta el momento son los eventos de cada objeto, internos al objeto, tanto disparados por el usuario como por el sistema. Por ejemplo el ClientStart, el Start, el Refresh, el Load, que son del sistema, o los eventos asociados a los controles de la User Interface.

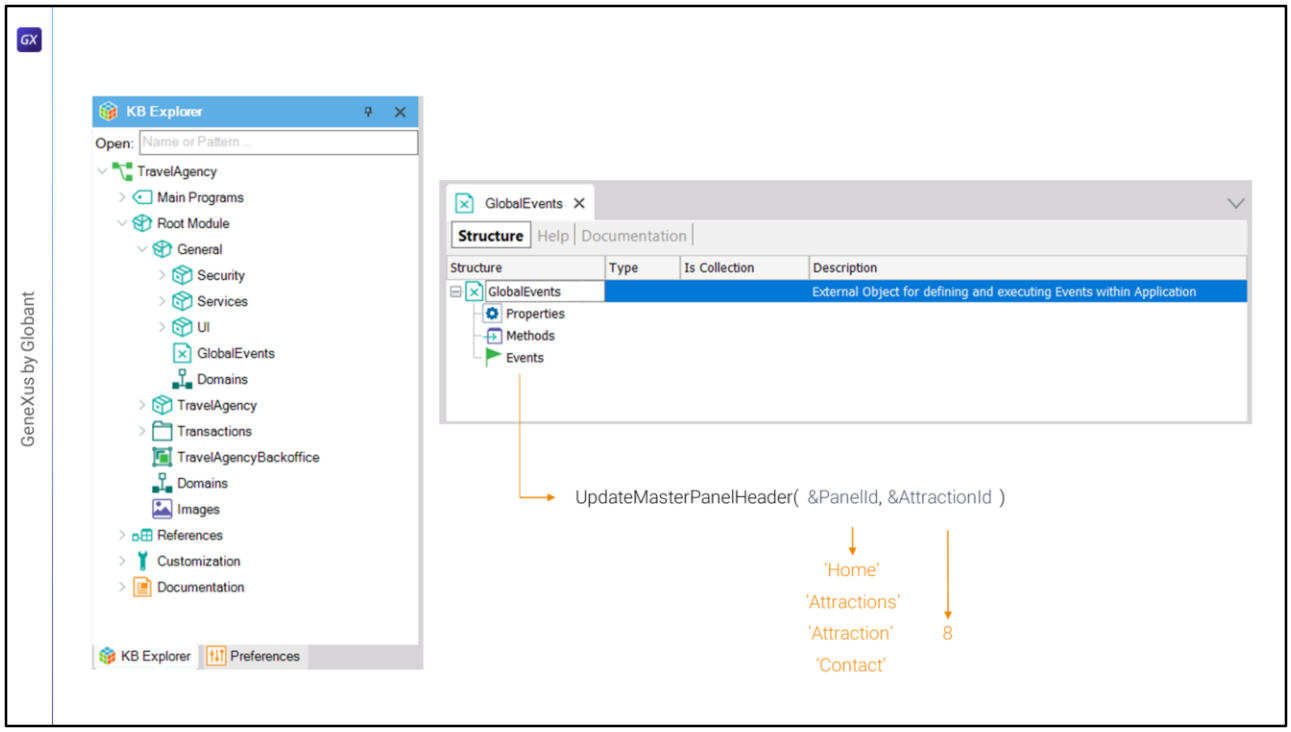
Pero esos son todos eventos locales, son disparados y manejados dentro del objeto que los define, internamente.



En cambio, los **eventos globales** valdrán para todos los componentes del contexto de un objeto que se esté ejecutando. En nuestro caso el Panel y el Master Panel se encuentran dentro de un mismo contexto de ejecución. Por lo que, si el Panel dispara un evento global, el Master Panel, que está en su mismo contexto de ejecución, podrá escucharlo y ejecutarlo. Esto mismo valdrá para componentes, como les voy a contar más adelante.



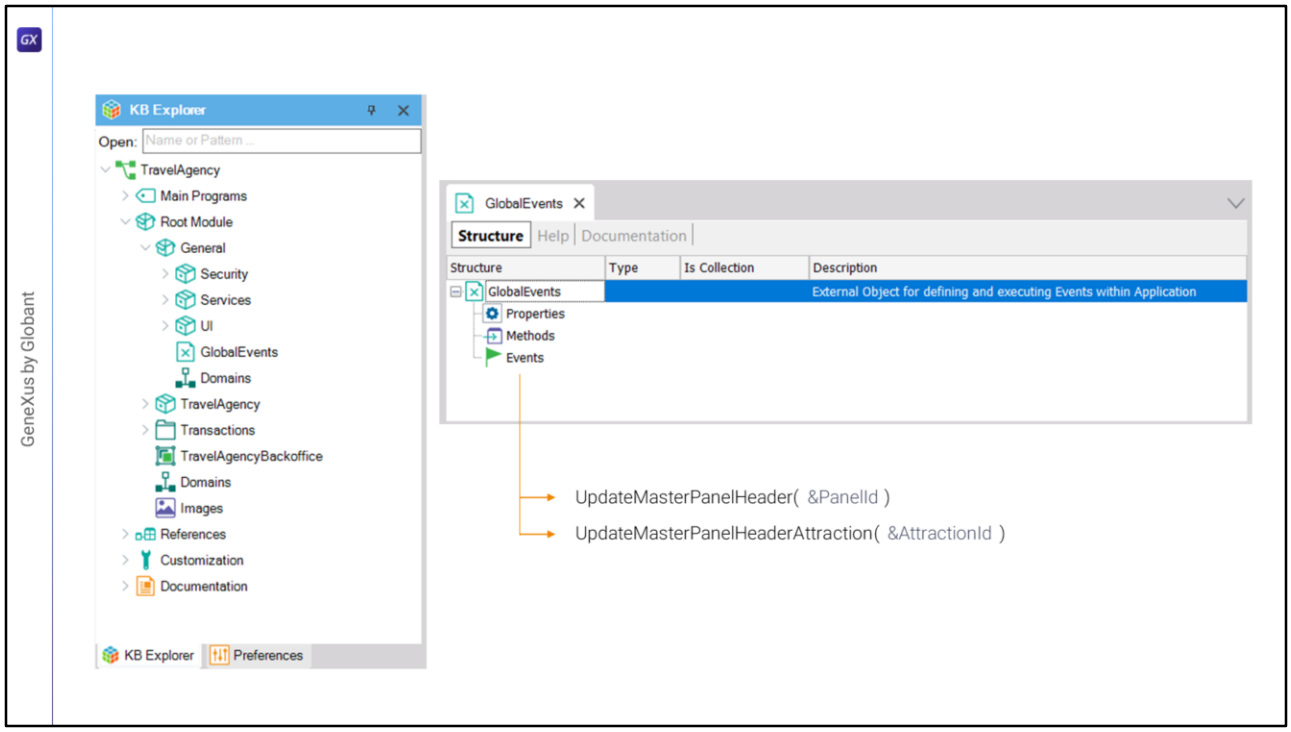
Y esto es lo que necesitamos. Que los paneles disparen un evento que sea escuchado y ejecutado por el Master Panel, a través del que puedan pasarle información al Master Panel.



Para ello es que toda KB disponibiliza dentro del módulo General un objeto GlobalEvents, válido para todas las plataformas (web y nativas).

Allí, bajo su nodo Events, es donde podremos definir eventos globales, para que puedan ser compartidos entre objetos en un mismo contexto de ejecución.

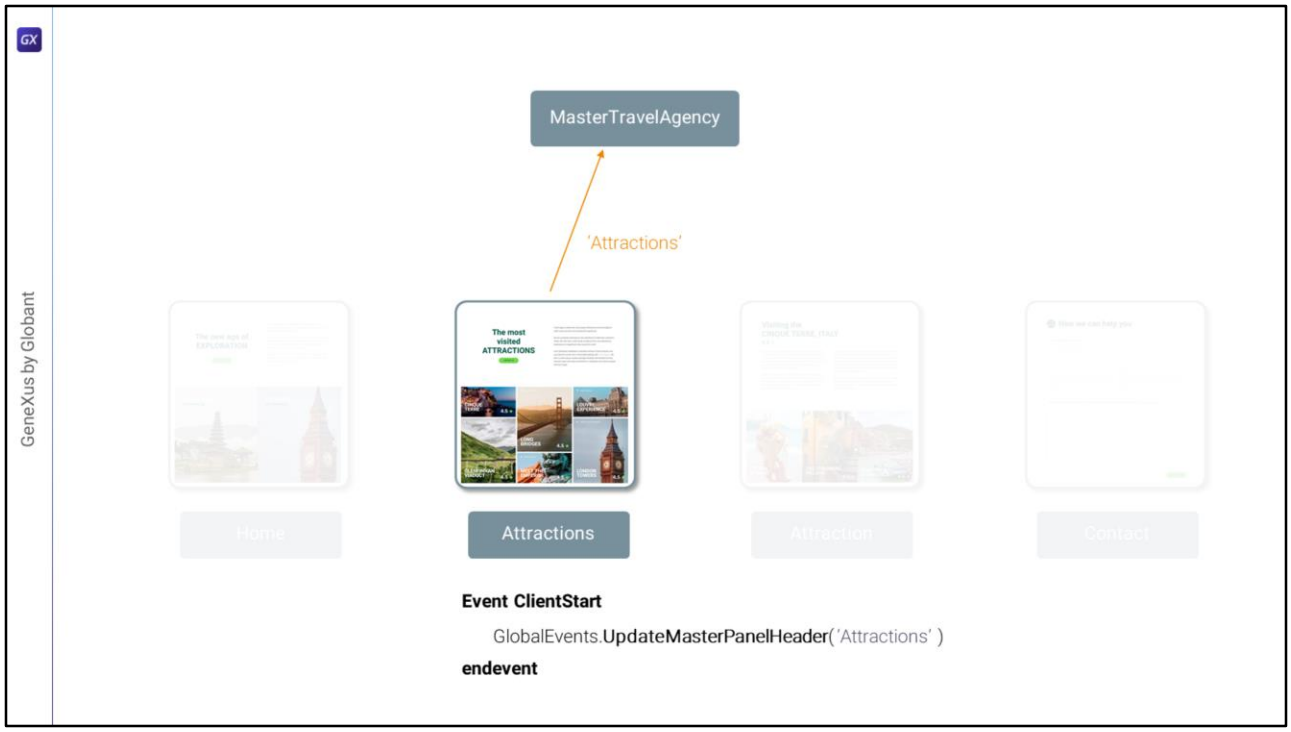
Para nuestro caso podríamos definir un único evento, que al dispararse (en nuestro caso desde los 4 panels) deberá enviar a quien escuche (en nuestro caso el Master Panel) **2 parámetros**: el primero para identificar al objeto que lo está disparando (será 'Home', 'Attractions', 'Attraction' o 'Contact'), y el segundo sólo tendrá sentido si el que lo está disparando es Attraction, porque es para identificar a la atracción turística.
Si los que lo están disparando son Home, Attractions o Contact, el valor que pasen para el segundo parámetro no será tenido en cuenta por el Master Panel.



La otra opción es diferenciar dos eventos, uno que sea el que se invocará desde Home, Attractions o Contact para identificarse, y el otro que será disparado únicamente desde Attraction, enviando el Id de la atracción.

Las dos soluciones (un solo evento, versus dos diferenciados) tienen pros y contras. Voy a elegir la segunda, con dos eventos, sólo para que vean que podemos agregar todos los eventos globales que deseemos a este objeto GlobalEvents.

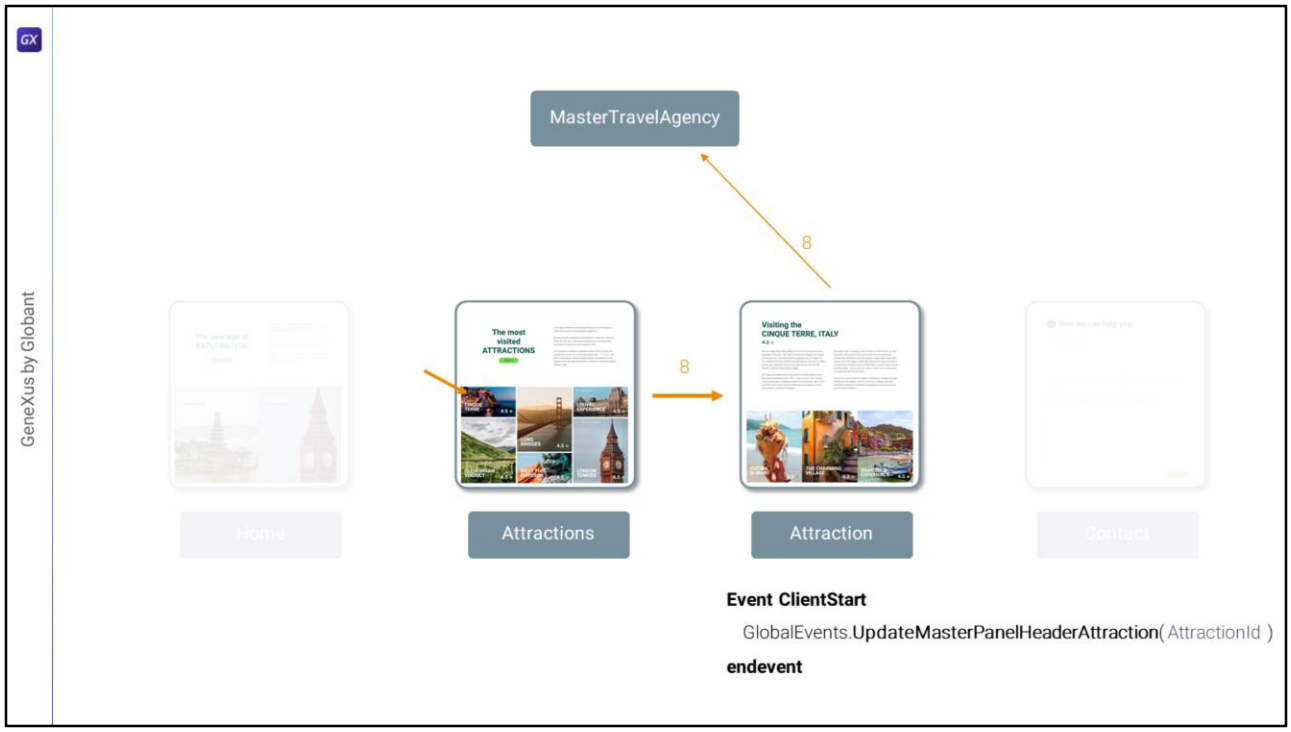
Entonces para el panel Home, en el ClientStart voy a disparar el evento UpdateMasterPanelHeader, pasándole por parámetro el valor ‘Home’.



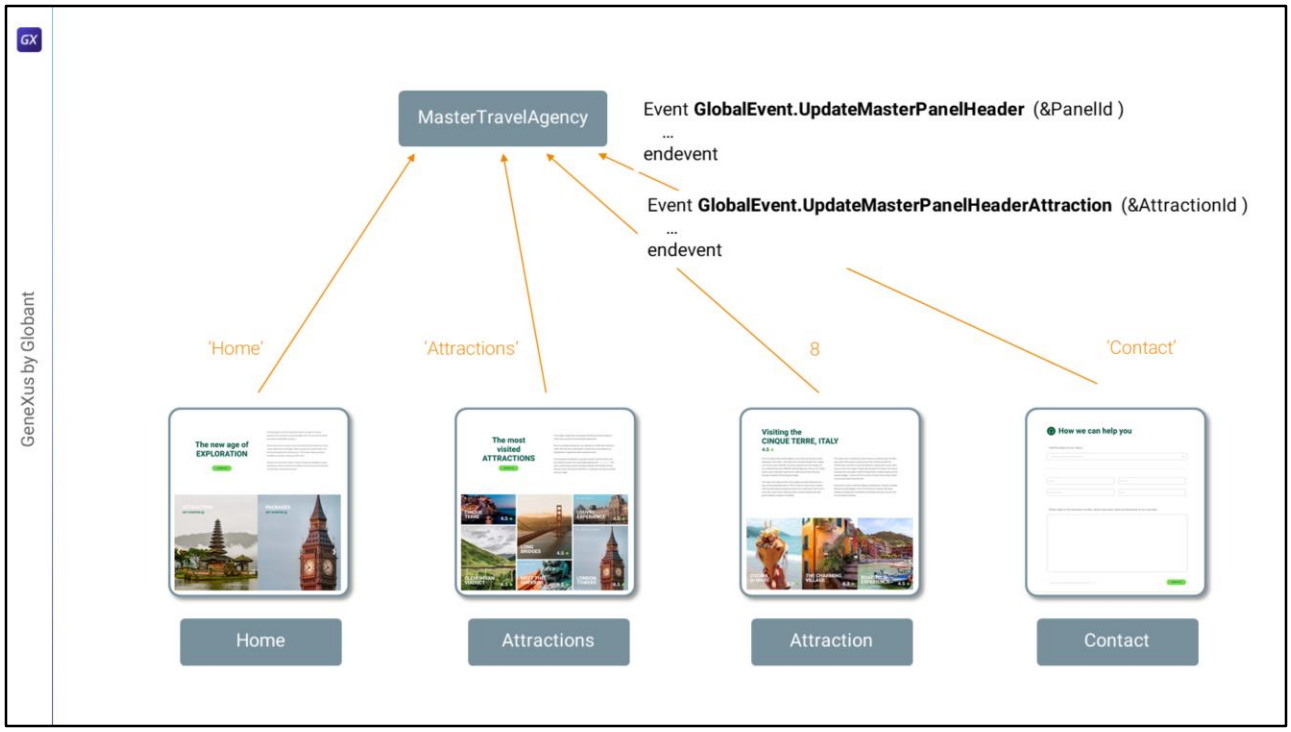
Y lo mismo se hará en el ClientStart de Attractions, pero pasándole, claro, ‘Attractions’ por parámetro…

…y en el de Contact, pero pasándole ahora ‘Contact’.

En el de Attraction se disparará el otro evento, el particular, UpdateMasterPanelHeaderAttraction…

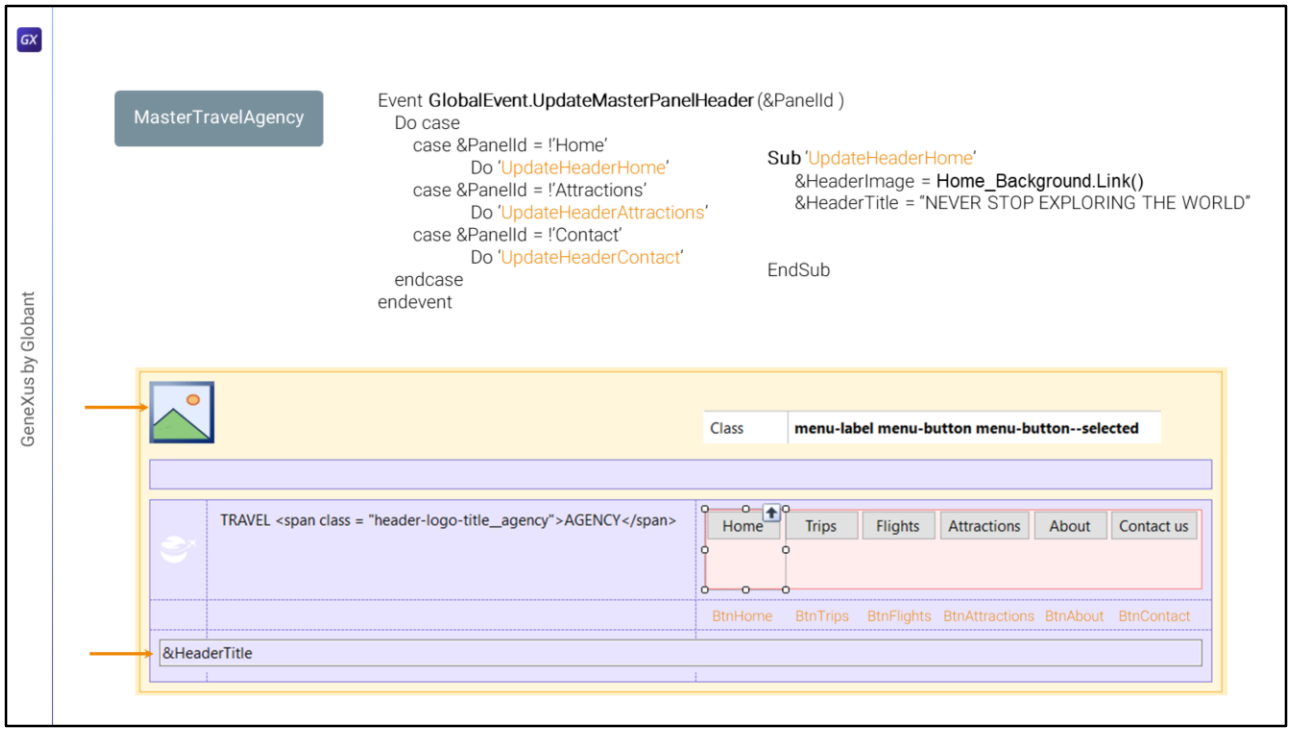


Y como el panel Attraction recibirá a su vez por parámetro el id de la atracción turística (que le será enviado desde aquí, cuando el usuario seleccione una atracción para ver su info, por ejemplo, la de id 8), ese mismo id de atracción será el que le enviará por parámetro a este evento global.



Luego, el Master Panel sólo deberá “escuchar” estos dos eventos, para recibir esa identificación y proceder.

Así, si se está ejecutando alguno de estos tres paneles, dispararán este evento.

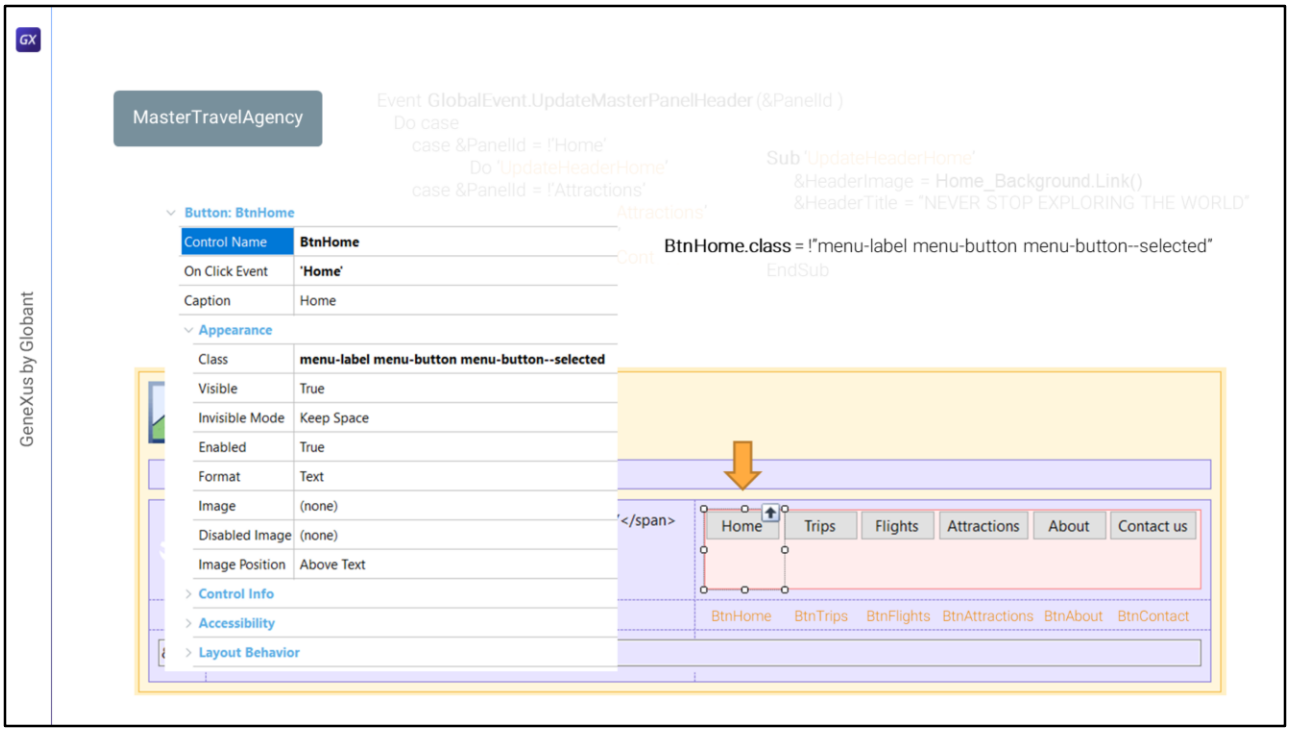


¿Y qué es lo que tiene que hacer el evento? En base a quién sea el que lo disparó… (este signo de exclamación antes de estos textos es para indicarle a GeneXus que esos textos son internos a la aplicación y, de traducirse la aplicación a otros idiomas, estos textos no deberán traducirse)… Bueno, decía, en base a quién sea el que lo disparó deberá actualizar el Header cargando las dos variables: &HeaderImage y &HeaderTitle con los valores que le correspondan para ese panel que lo disparó… y para los botones, deberá dejarlos todos deseleccionados salvo uno, el que corresponda al panel.

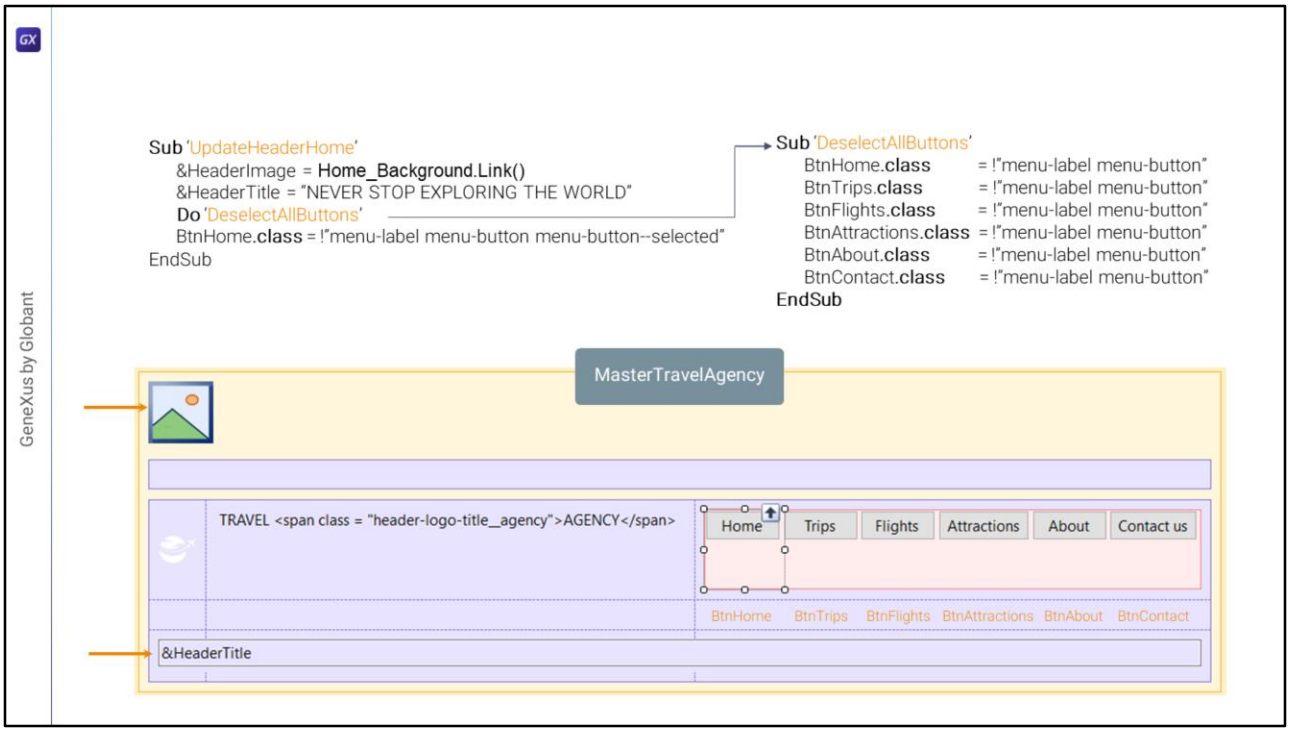
Esto que estamos escribiendo acá son invocaciones a subrutinas, que son porciones de código locales al objeto, que se pueden aislar para reutilizarlas o para hacer más comprensible la semántica del código.

Dentro de cada una de esas subrutinas, por ejemplo veamos la de Home… se actualizará, entonces, la variable de la imagen, la del título… y se dejará el menú con la opción seleccionada correcta. ¿Cómo hacemos esto último?

Si a los controles botón los nombramos de esta manera… recordemos del video anterior que teníamos dos clases para todos los botones: la menu-label para la tipografía y la menu-button para dar el borde de abajo y arriba transparente, ¿se acuerdan? (podríamos haber unido las dos en una sola, pero bueno, no lo hicimos) y una tercera clase para lograr el indicador verde debajo, que era la “menu-button--selected”, que en el video anterior se la habíamos dejado asociada de manera estática al botón Home.

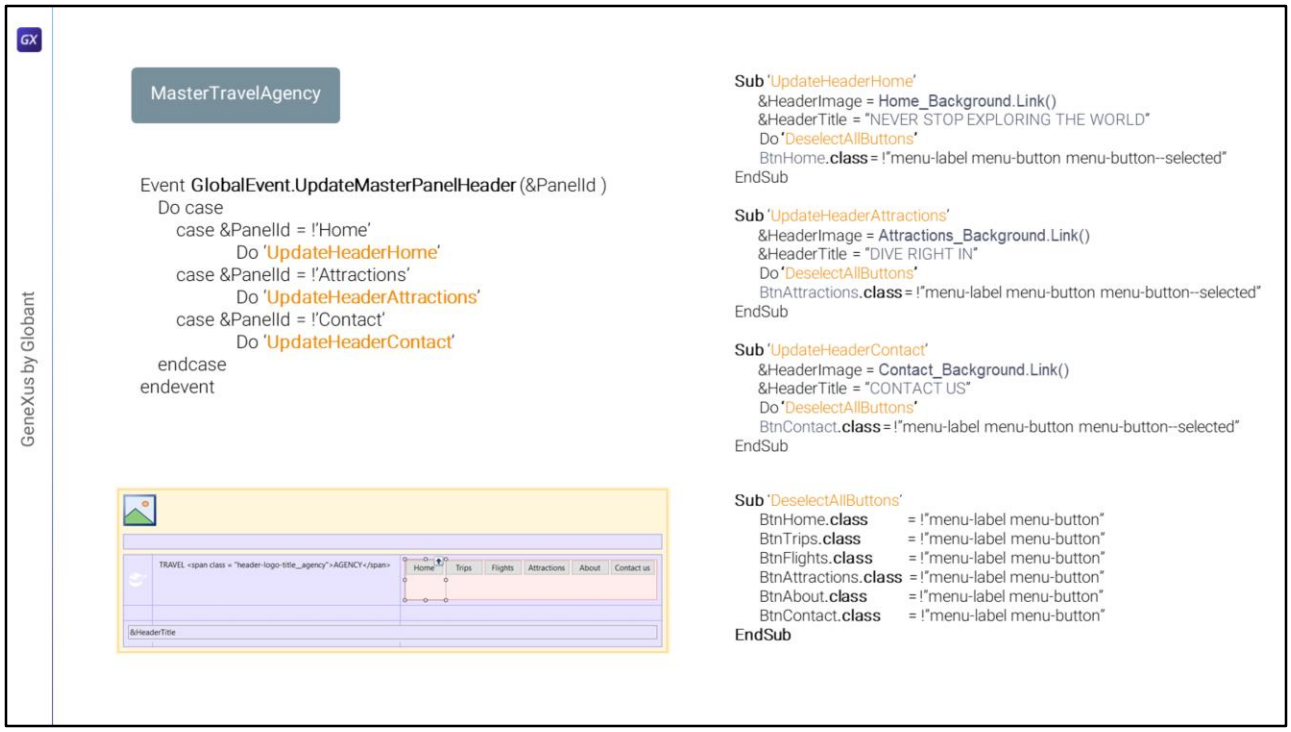


Las clases se pueden asignar a los controles tanto en forma estática, a través de la propiedad del control, como dinámica, en cualquier lado donde se acepte código, como en los eventos.



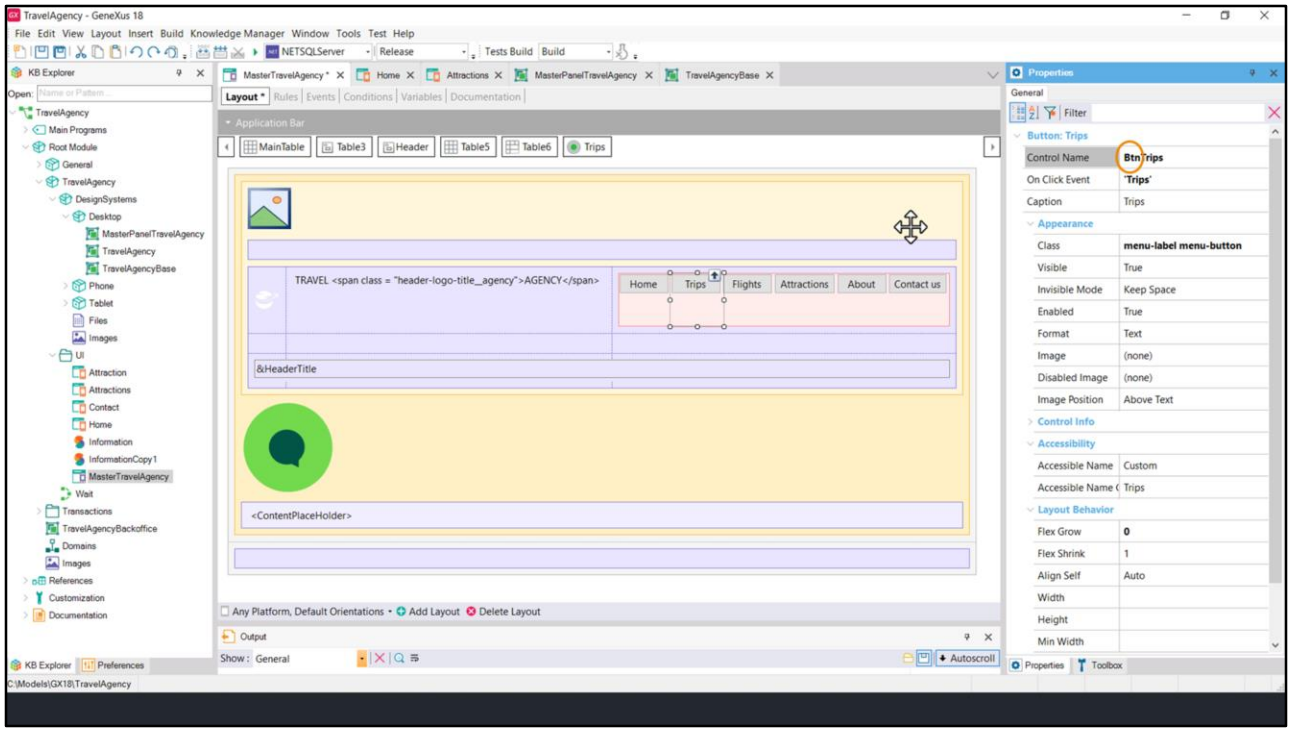
De esta manera, podríamos escribir una subrutina “DeselectAllButtons”, que vemos que lo que hace es asignarle las dos clases, menu-label y menu-button, a todos los botones del menú.

Entonces lo que haríamos en esta subrutina de la que partimos sería primero invocar a esta otra para dejar todos los botones como deseleccionados. Y luego simplemente asignarle al botón que corresponde, que en este caso es el de nombre BtnHome, sus clases, que son las mismas que las de los otros, más la menu-button--selected.



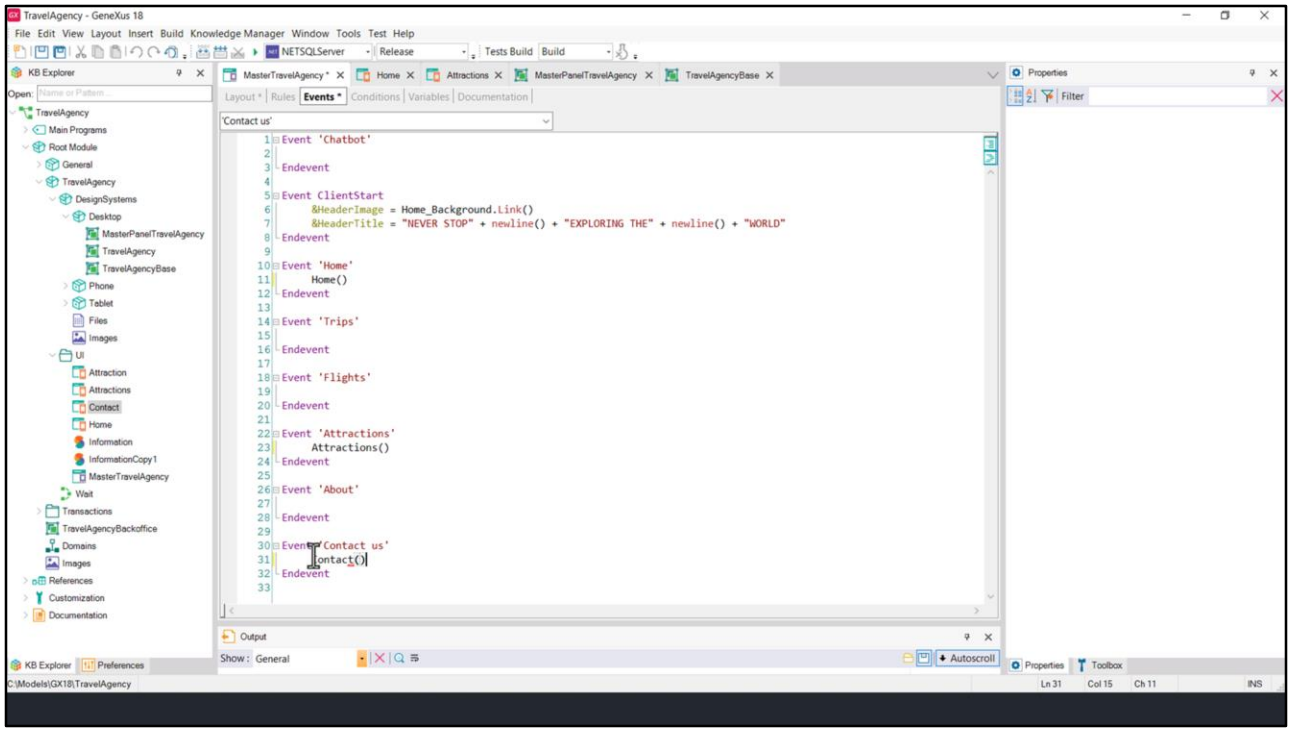
Así nos quedará entonces la solapa de eventos del Master Panel para implementar correctamente el Header cuando el Master Panel esté siendo llamado por los paneles Home, Attractions o Contact. Vean que las subrutinas para los dos últimos tienen exactamente la misma lógica que vimos para Home: cargan la imagen que corresponde, el título y deseleccionan todos los botones, para luego dejar seleccionado sólo el que corresponde.

Nos falta ahora implementar la carga correcta del Header cuando quien llama es Attraction, pero antes llevemos todo esto a GeneXus, para ir sobre seguro.

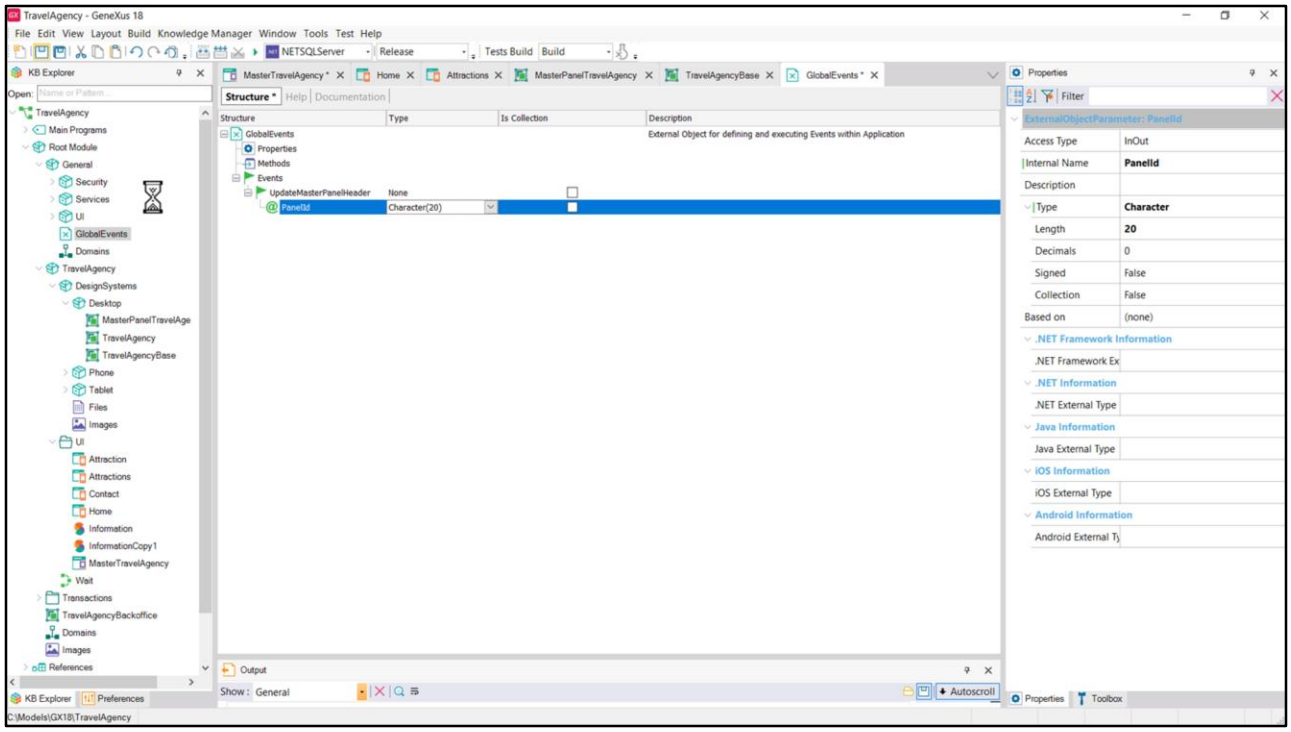


Aquí tenemos el Master Panel. Recordemos que por un bug en el editor quedaron invisibles los botones. Por el momento, para volver a tenerlos visibles y poder operar con ellos, modifiquemos esta propiedad, que era la que se relacionaba con el bug. Después la dejaremos como debe ir, con el valor Center.

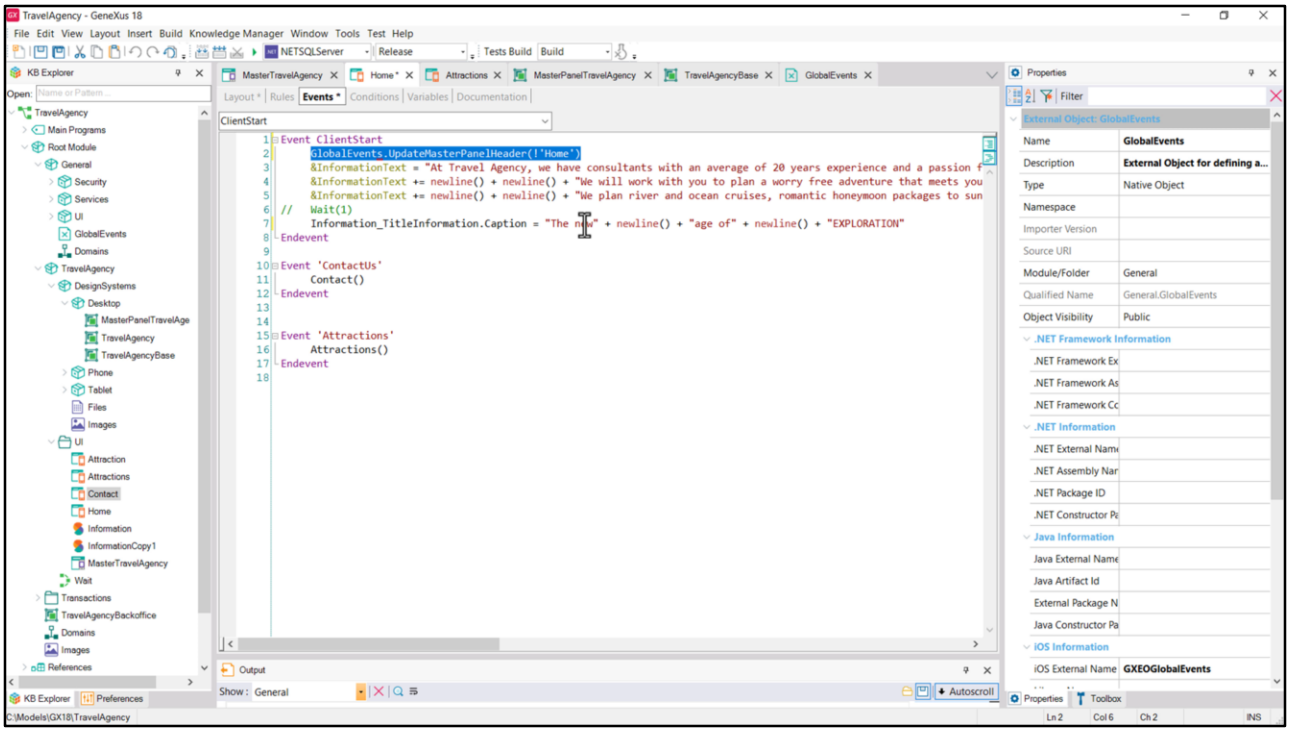
Aprovechemos y cambiémosle el nombre a los botones para antecederlos con “Btn”, así nos quedará más claro su uso luego en el código.



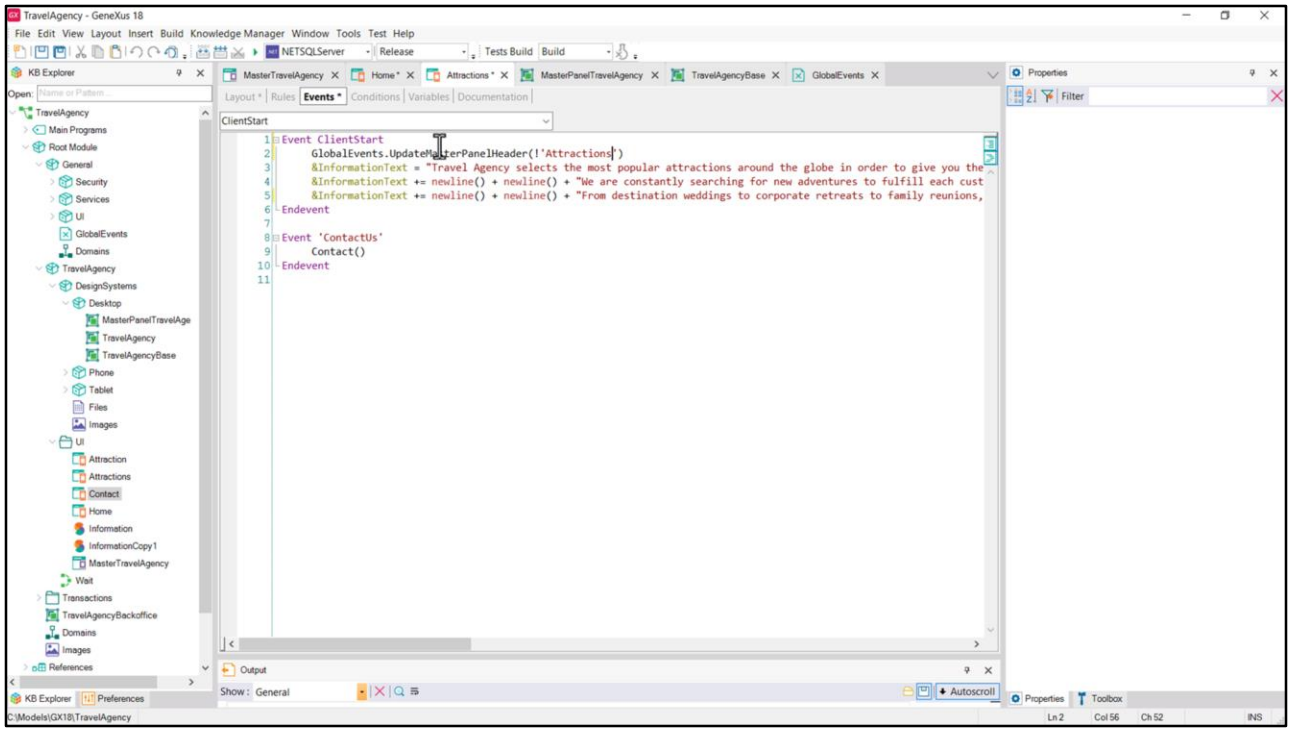
Empecemos por realizar las invocaciones en cada botón a los paneles. Así, cuando se presione el botón de Home deberemos invocar al panel Home. Cuando se presione Attractions al de Attractions y en el de Contact us al panel Contact. Los otros paneles por ahora no los tenemos ni siquiera diseñados.



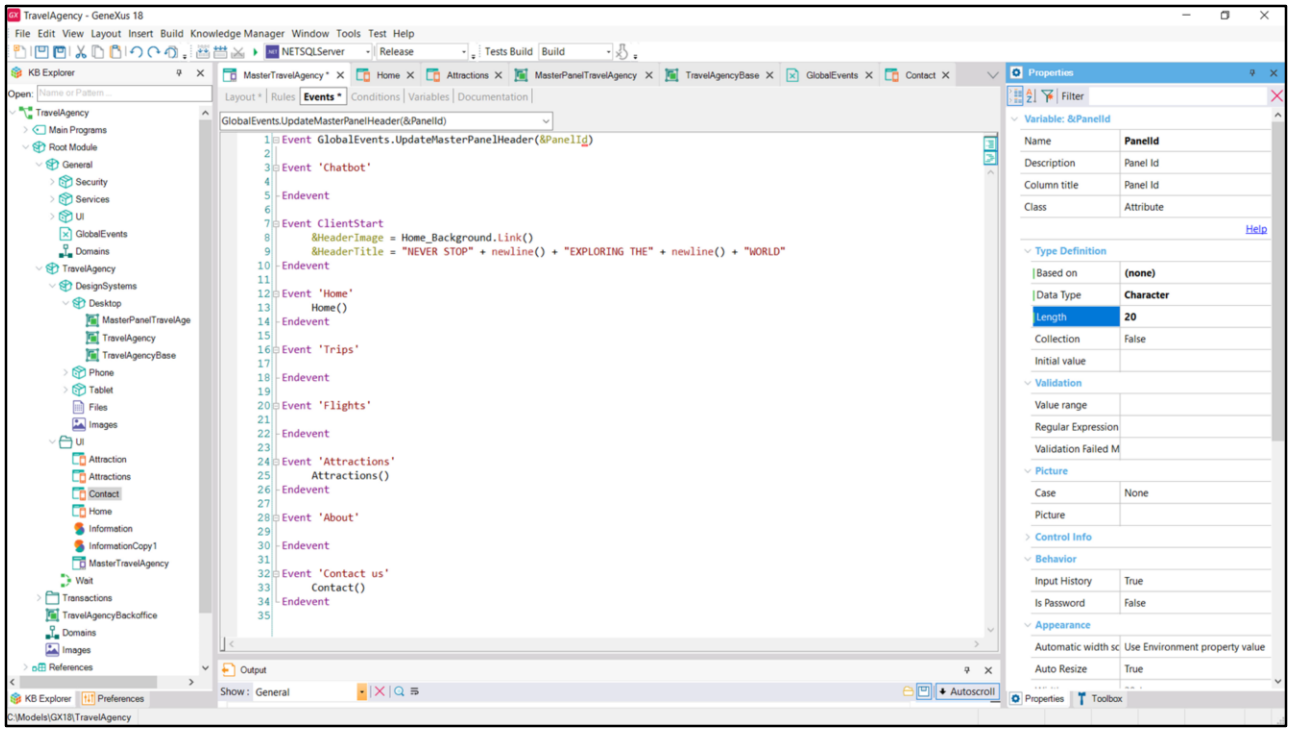
Ahora editemos el objeto GlobalEvents. Agreguemos el evento UpdateMasterPanelHeader, con variable &PanelId de tipo Character.



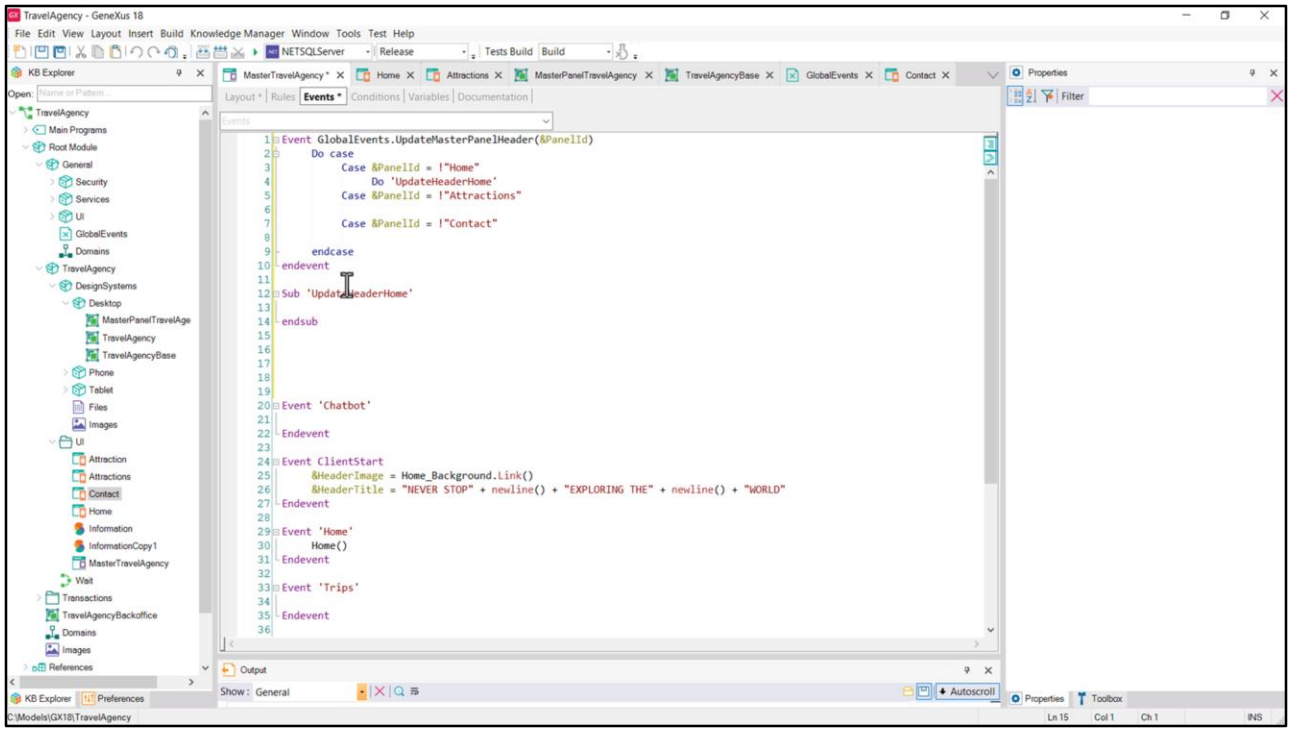
Vayamos a Home y en el ClientStart lo disparamos. Voy a ponerlo mejor al principio del código para que quede más claro.



Y lo mismo hacemos en Attractions, cambiando aquí por ‘Attractions’. Y en Contact.

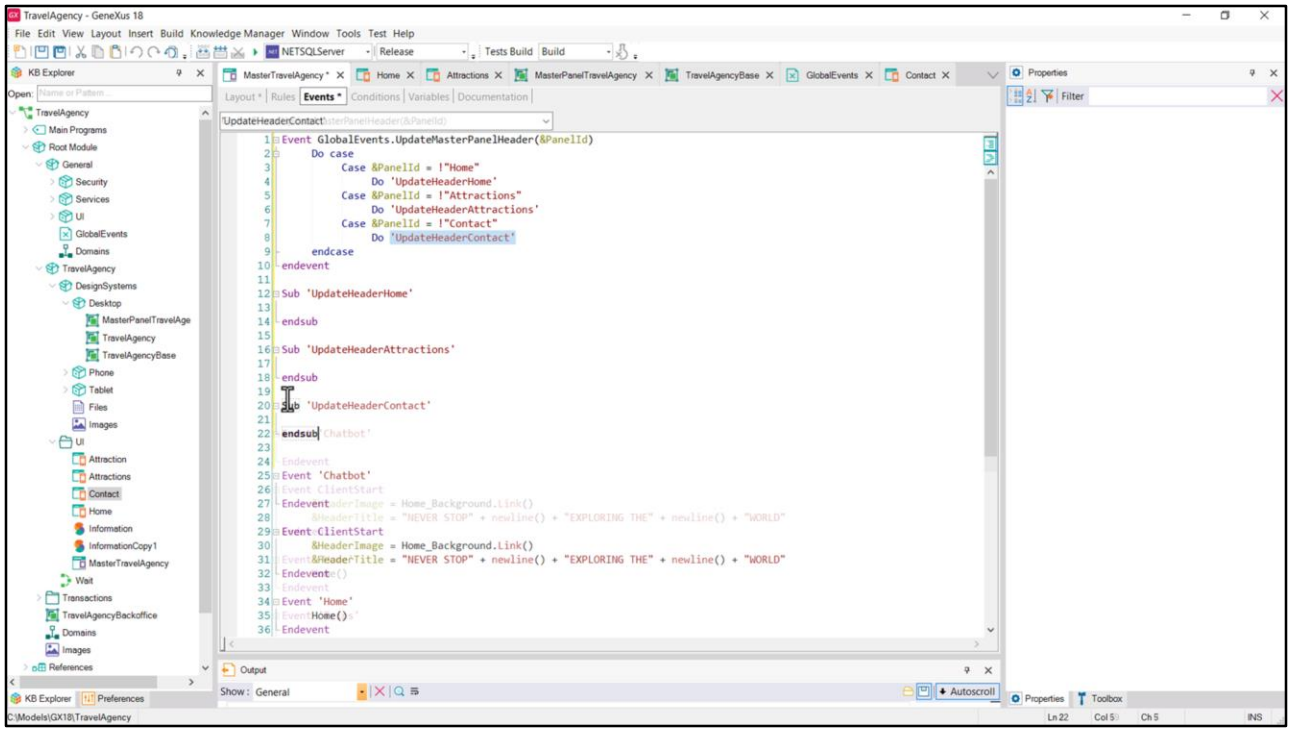


Ahora programaremos el evento global en el Master Panel… Definamos la variable… &PanelId… no la queremos basada en el dominio Id sino que queremos que sea de tipo Character.

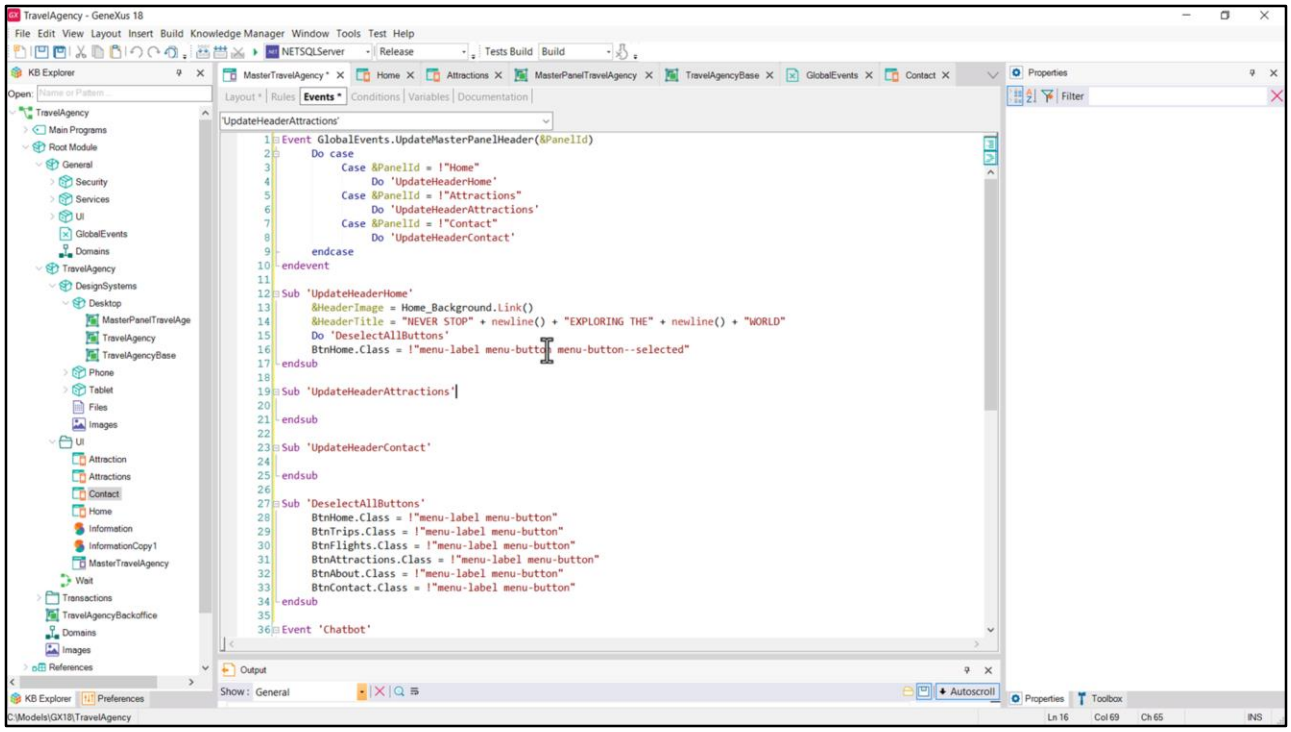


Bien, agregamos el comando **Do case**, para poder tomar las distintas acciones en base al valor de la variable… Si es ‘Home’ se hará una cosa, si es ‘Attractions’ otra, si es ‘Contact’ otra.

¿Qué se hará si es ‘Home’? Invocar a esta subrutina, que tenemos que definir… La podemos especificar en cualquier lugar de esta solapa de eventos, que como sabemos es declarativa, no importa el orden en el que definamos los eventos y las subrutinas.

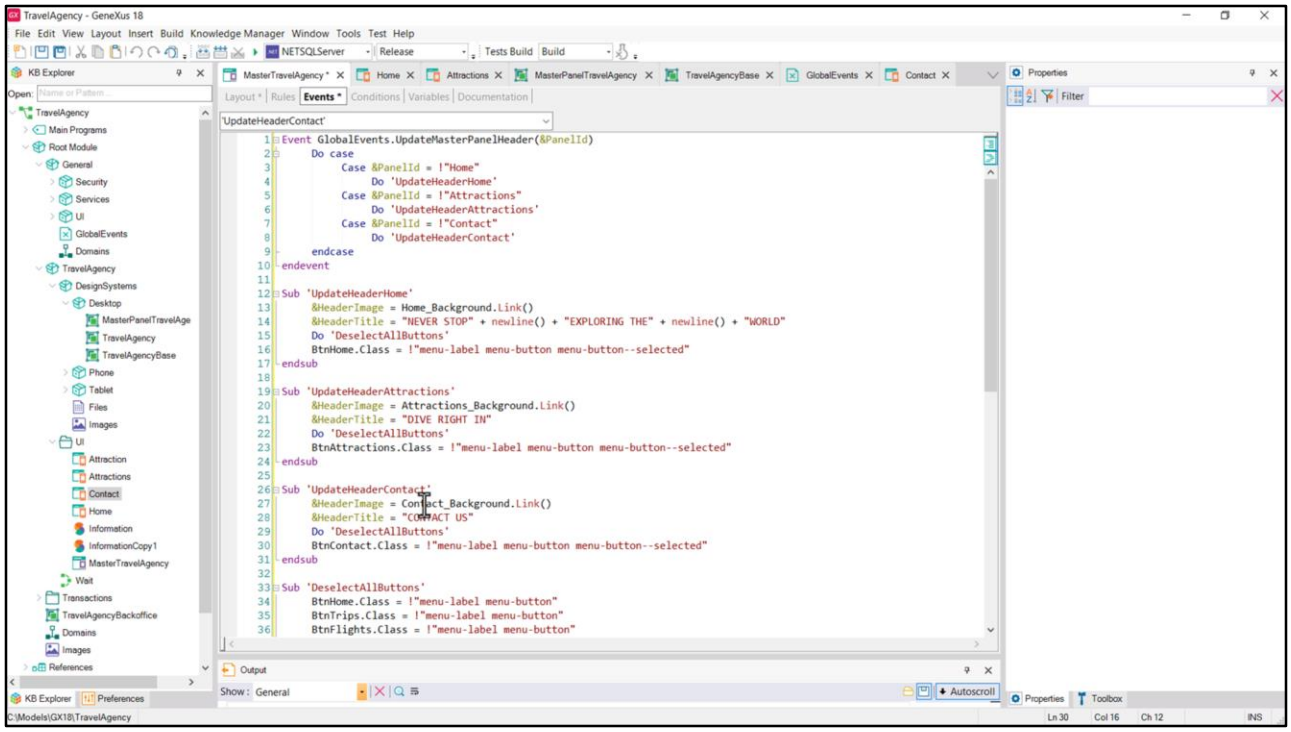


Si el valor que toma la variable es ‘Attractions’ llamaremos a esta otra subrutina. Y si es ‘Contact’ a esta otra.



Programemos la primera. A la variable &HomeImage le asignamos la imagen que habíamos insertado en la KB en la etapa de preparación, método link.
Al título este de aquí.
Luego invocamos a la subrutina DeselectAllButtons que definiremos luego. Y por último al botón Home le asignamos las 3 clases.

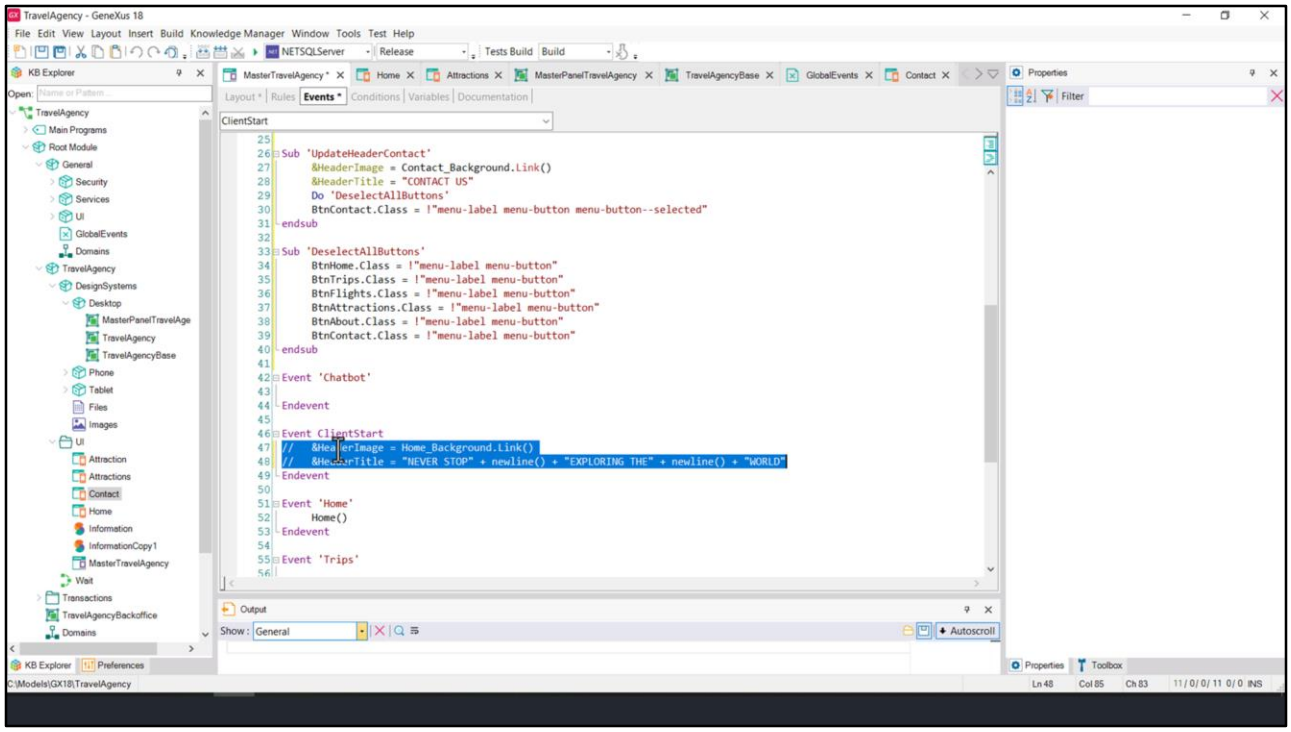
Ahora especifiquemos la subrutina DeselectAllButtons, en la que a todos los botones del menú les asignamos las primeras dos clases.



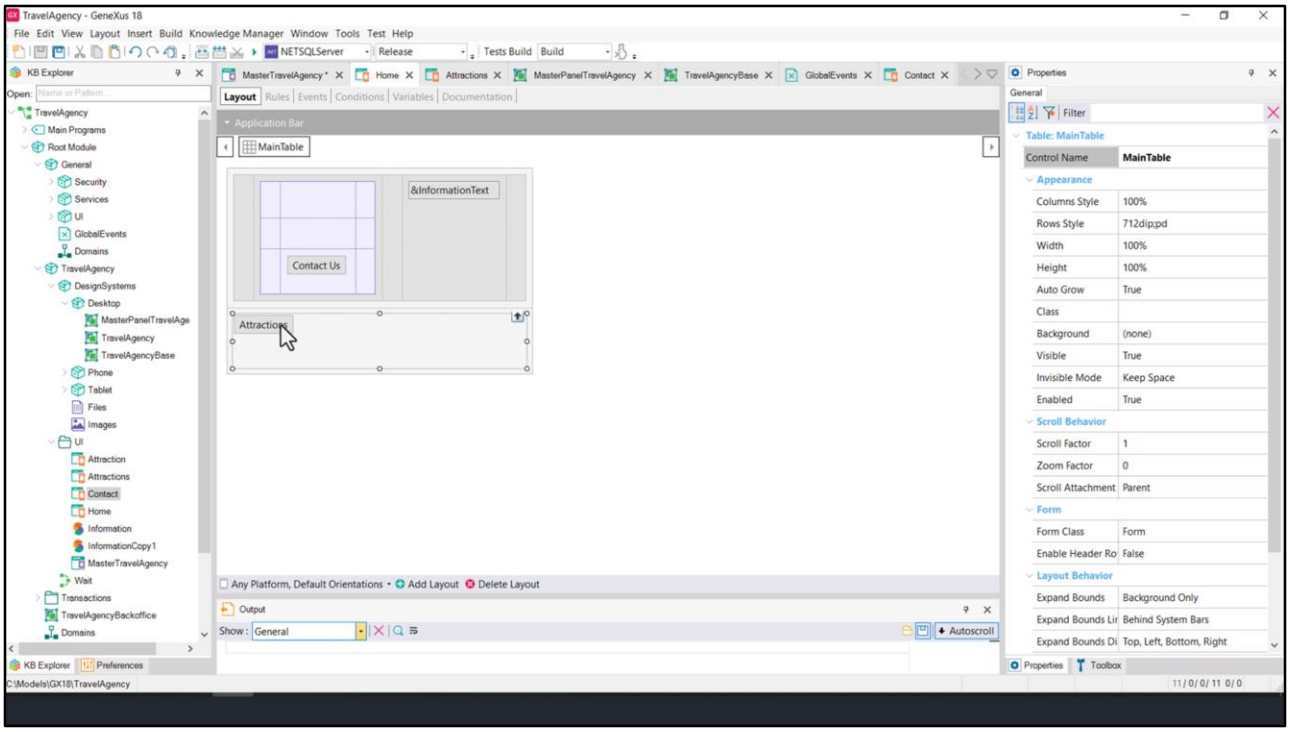
Análogamente, programamos las otras dos subrutinas…
Aquí la imagen se llamaba así… y el texto era este…(lo extrajimos de Figma)… y el botón aquí es el de Attractions.

Y para Contact también hacemos las sustituciones necesarias.

Al grabar nos da un error… no está entendiendo el nombre BtnContact. Si venimos… es que se llamaba así. Vamos a quitar este final. Ahora sí.



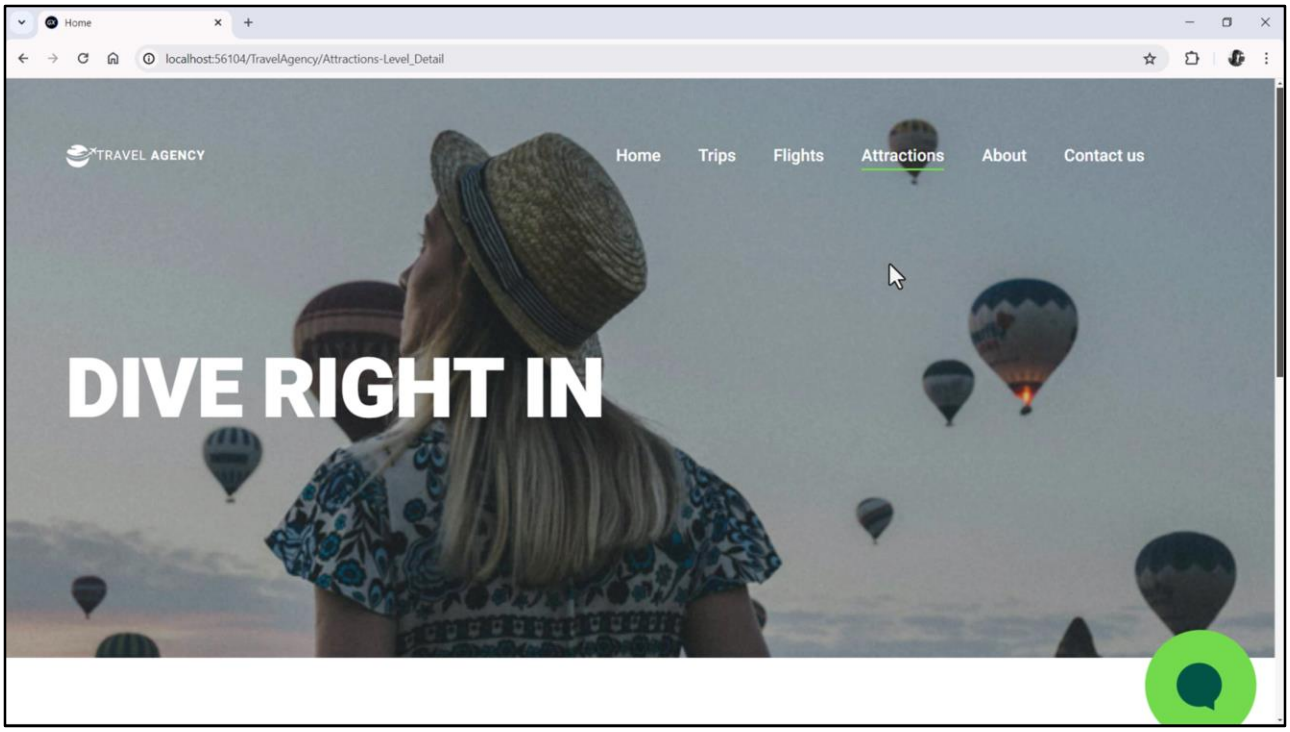
Esta carga del ClientStart del Master Panel ya no será necesaria.



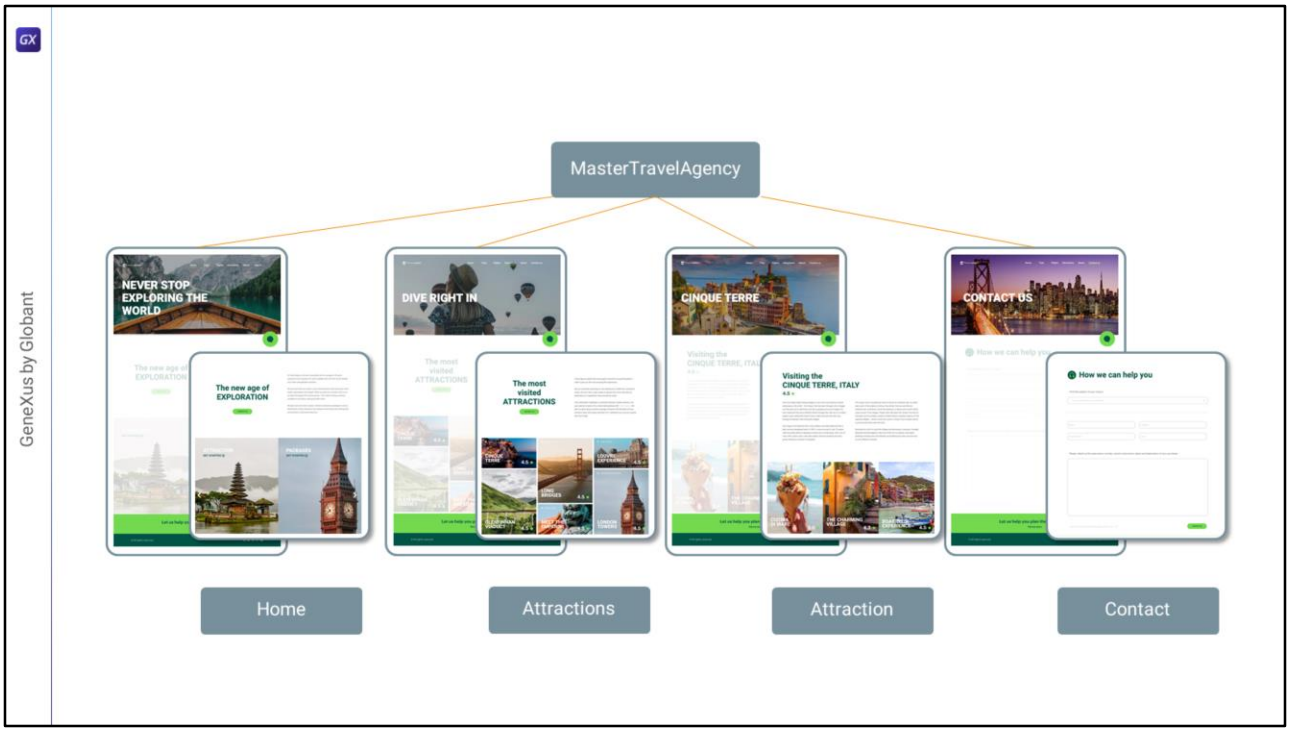
Así como tampoco este botón para llamar al panel de atracciones.

Aquí observemos todo… está todo bien.

Volvamos a colocar la propiedad Align Items del flex en Center. Y Ejecutemos.

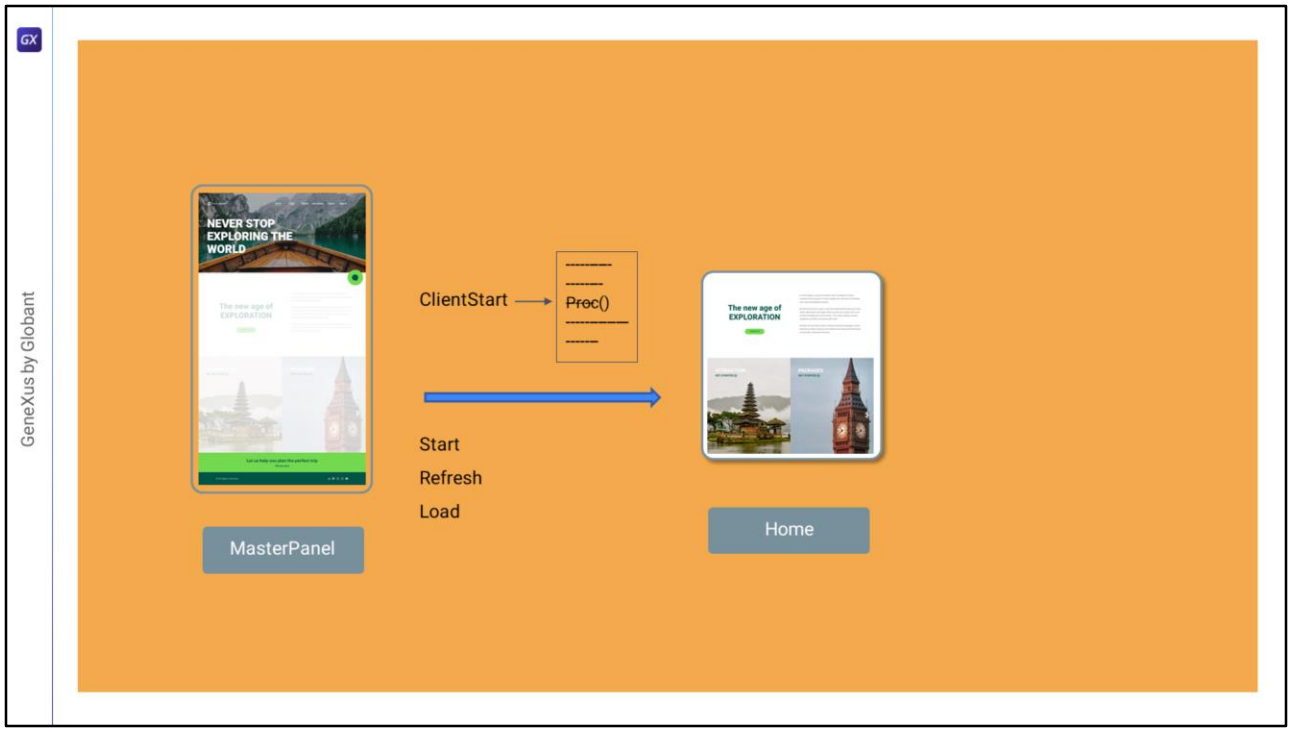


Vemos perfecta la página Home… y si cliqueamos aquí, perfecta la de atracciones, y aquí la de contacto. Volvemos a Home… está todo perfecto.



Por el momento (y esto está en proceso de modificarse, así que dependerá de cuándo veas este video si esto sigue siendo así o no), cada vez que se ejecuta un panel se vuelve a ejecutar su Master Panel, no importando para nada si entre ejecuciones de paneles el Master Panel es el mismo. Es decir, se construye y destruye entre navegaciones, cada vez.

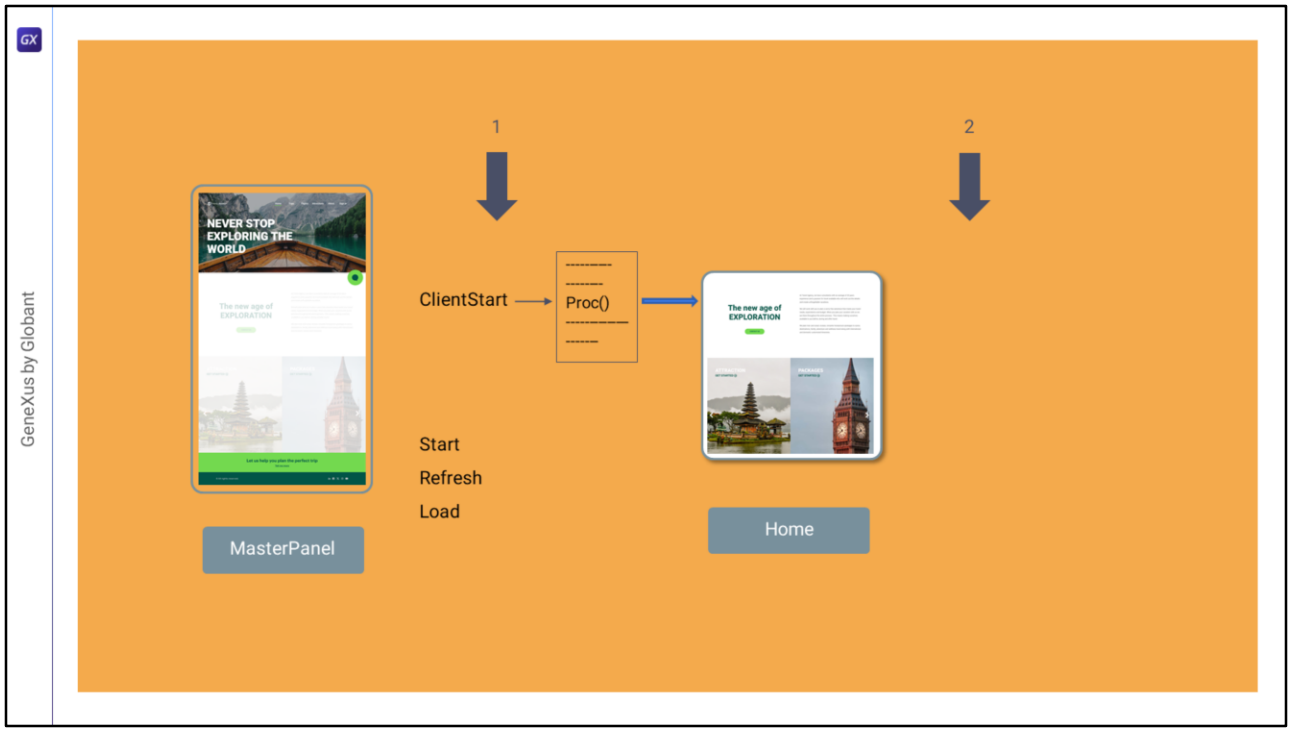
Eso ciertamente no es lo más eficiente del mundo y puede traer complicaciones de parpadeo, por ejemplo.



Para entender un poco mejor todo esto, les voy a contar cómo funciona por el momento.

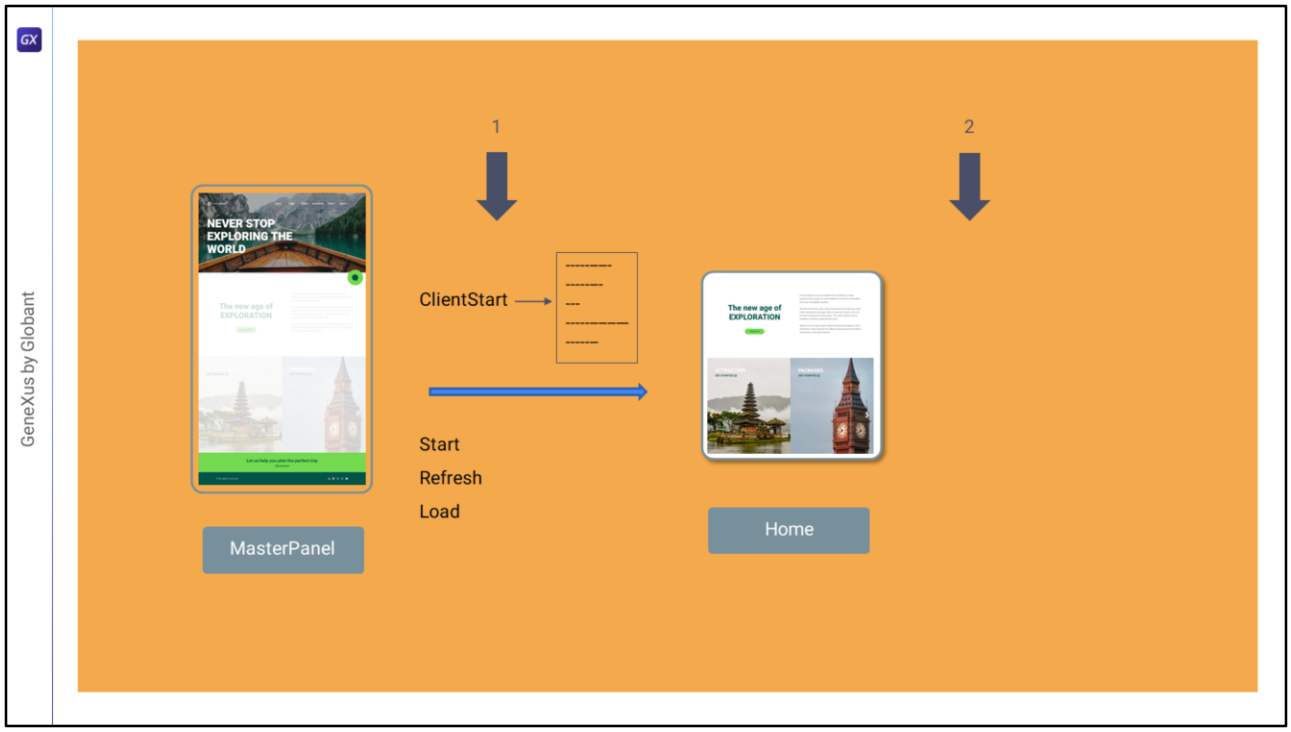
Al lanzar la ejecución del panel Home, por ejemplo, el orden de ejecución será el siguiente:
Primero se construye el Master Panel, donde se asignan todas las propiedades estáticas de los controles del layout (las que se indicaron en su momento en la ventana de propiedades de cada control; son las que solemos llamar propiedades en tiempo de diseño), y a continuación se ejecuta el evento ClientStart.
Si el evento ClientStart tiene alguna invocación a un procedimiento por ejemplo, que necesariamente deberá resolverse en el servidor, o a un Data Provider, se dispara su ejecución de manera asincrónica, y se renderiza por primera vez el Master Panel con lo que se tiene hasta el momento hasta aquí. Es decir, no se espera a que este Proc termine para seguir y recién al final del evento renderizar. Se renderiza y después se sigue.
En cambio si dentro del ClientStart no hay ninguna invocación asincrónica, entonces ahí sí se renderiza por primera vez recién cuando acaba el ClientStart.
Cualquiera sea el caso, tras la primera renderización del Master Panel ya se inicia la construcción del Panel proyectado en el ContentPlaceHolder, es decir, ANTES de ejecutarse los eventos Start, Refresh y Load.

Ya sea porque estamos aquí, ya sea porque estamos aquí



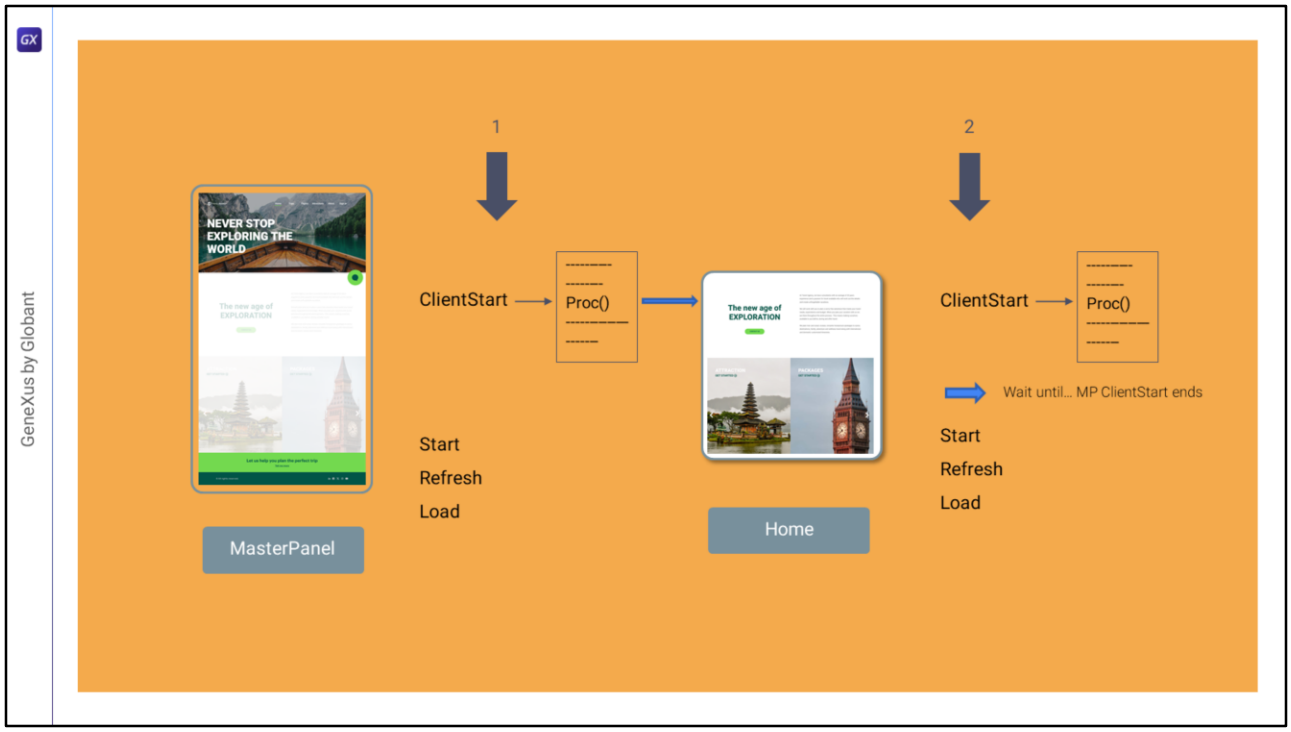
Se abre, entonces, otra línea de ejecución concurrente.

En la que estábamos, aquí ya se había renderizado por primera vez el Master Panel y lanzado la construcción del Panel. Cuando el Proc termine, continúa la ejecución del resto del ClientStart, y si hubo en el pPoc o aquí algún cambio en la User Interface se vuelve a renderizar. Luego se ejecutan los demás eventos del sistema (por supuesto, renderizando si introducen cambios en la UI).



Y si no había Proc ni nada asincrónico, entonces aquí ya se había renderizado el Master Panel y lanzado la construcción del Panel, por lo que ahora se ejecutan los demás eventos del sistema.

¿Y qué pasa en la línea concurrente?

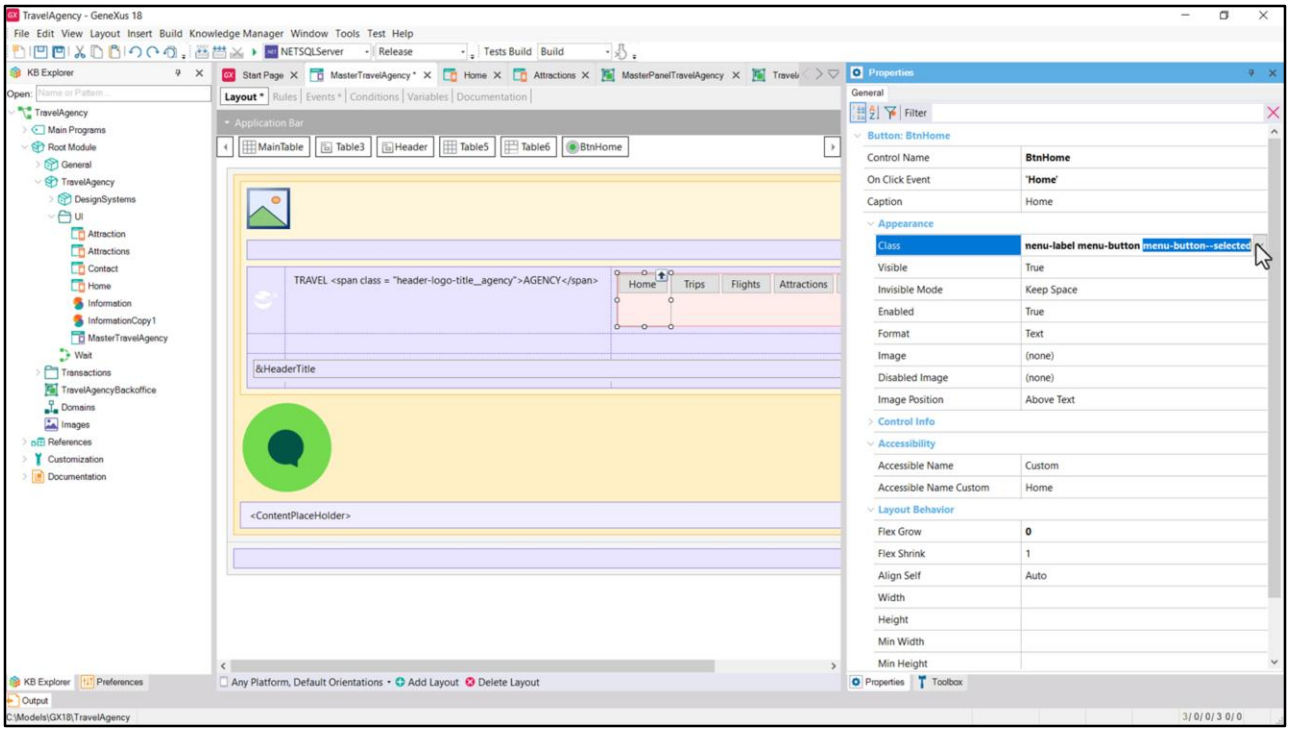


Veámoslo en el primer caso:

- Se construye la User Interface del panel en base a las propiedades definidas en Tiempo de Diseño (es decir, las estáticas),
- y se empieza a ejecutar el evento ClientStart, que, otra vez, si tiene alguna ejecución asincrónica entonces en ese momento se renderiza el panel, y se espera a que finalice la ejecución asincrónica. Cuando ésta termina, se continúa con la ejecución del ClientStart y al final, si es necesario, se renderiza nuevamente.
- Si no había ejecución asincrónica entonces se renderiza por primera vez el panel al finalizar el evento ClientStart.
- Y aquí presenten atención: si el ClientStart del Master Panel no finalizó cuando éste sí lo hizo, entonces aquí se espera hasta que lo haga, para recién luego empezar a ejecutar el Start y luego el Refesh y luego el Load.

En definitiva, el Start del Panel no comienza hasta que no haya terminado el ClientStart del Master Panel.

Sigamos ese orden en detalle para nuestro caso, prestando atención a qué pasa con la imagen y el título del Header y los botones del menú.

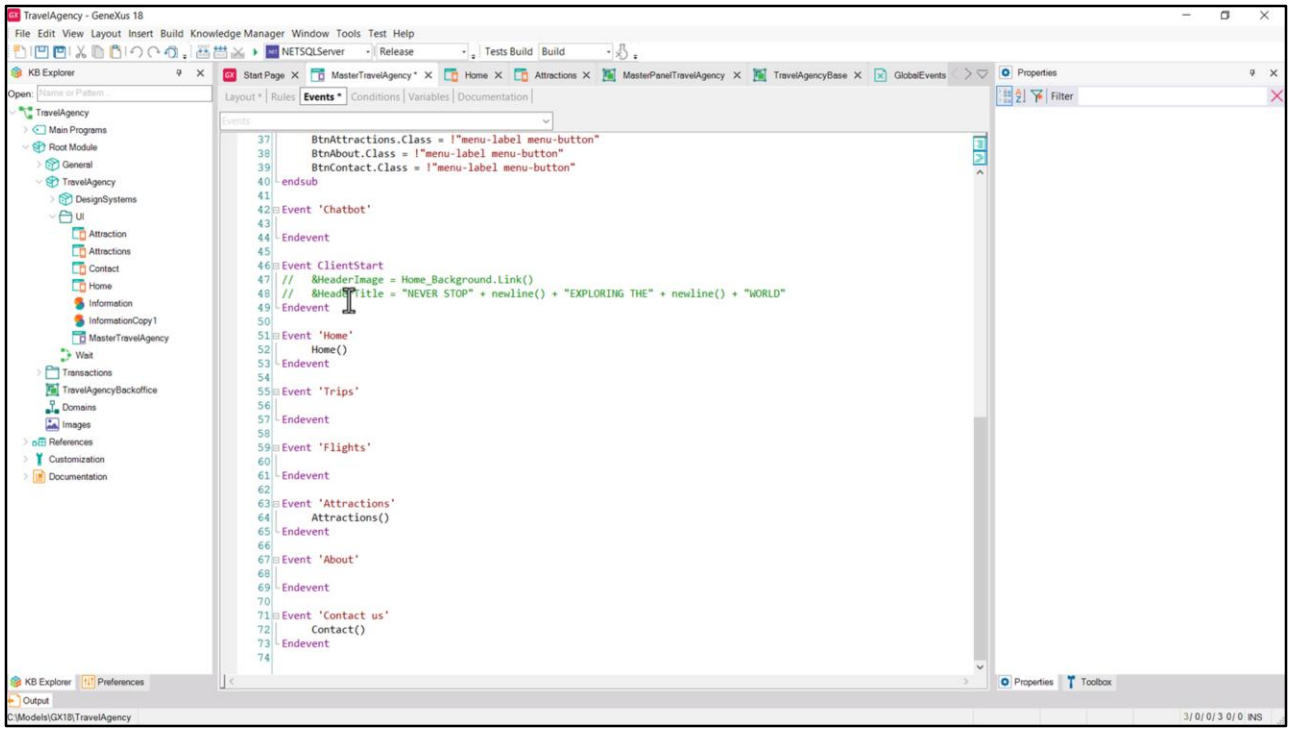


Entonces, se empieza por el Master Panel, decíamos, y lo primero que se hace es aplicar las propiedades estáticas.

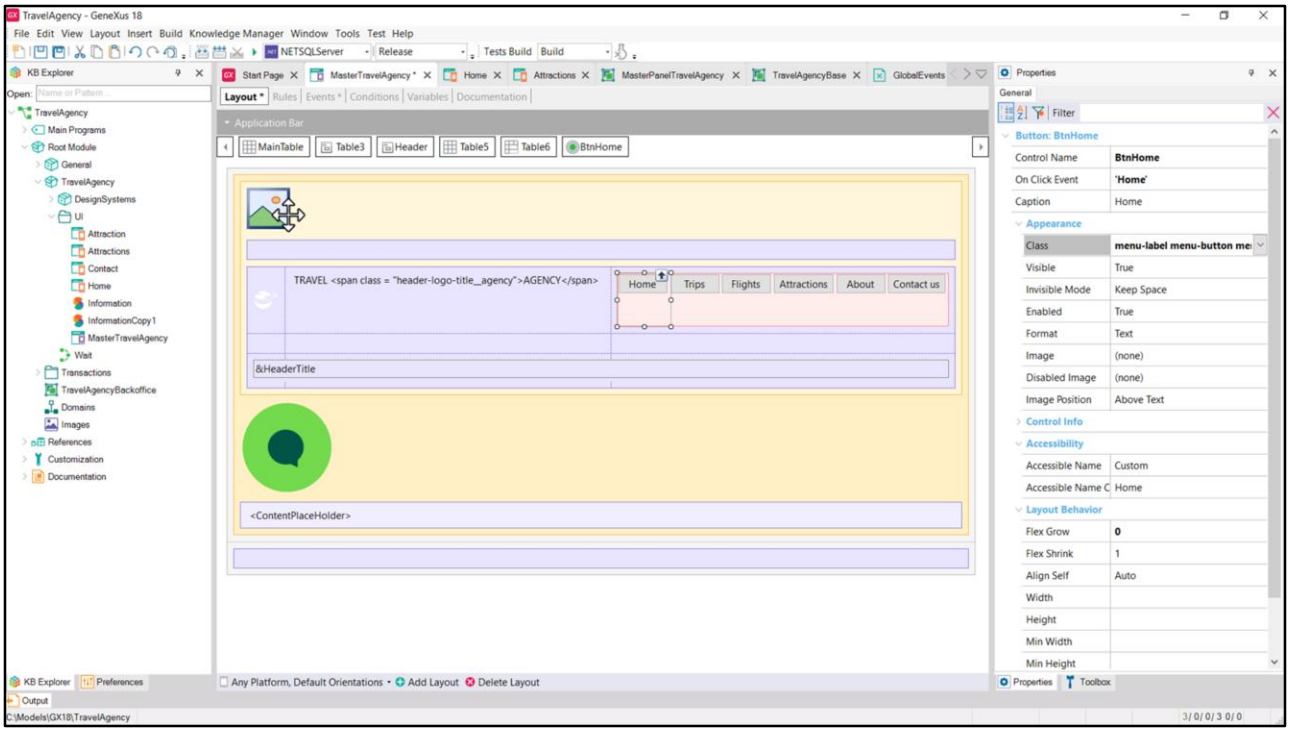
Entonces las variables de imagen y título estarán vacías en contenido, van a tener sus propiedades, las propiedades, por ejemplo, las de las clases.

Y el menú también, ya tendrá asociadas a cada botón todas estas clases. En particular, observemos que para el botón del Home tenemos también la clase seleccionada, que habíamos definido en el video anterior (lo habíamos dejado estático porque todavía no habíamos empezado a implementar toda esta cuestión dinámica de poder cambiar de pantalla, ¿no?, de acuerdo al menú). Tendremos que quitarla, claramente. Pero por ahora quiero dejarla para que pensemos por qué habría que quitarla.

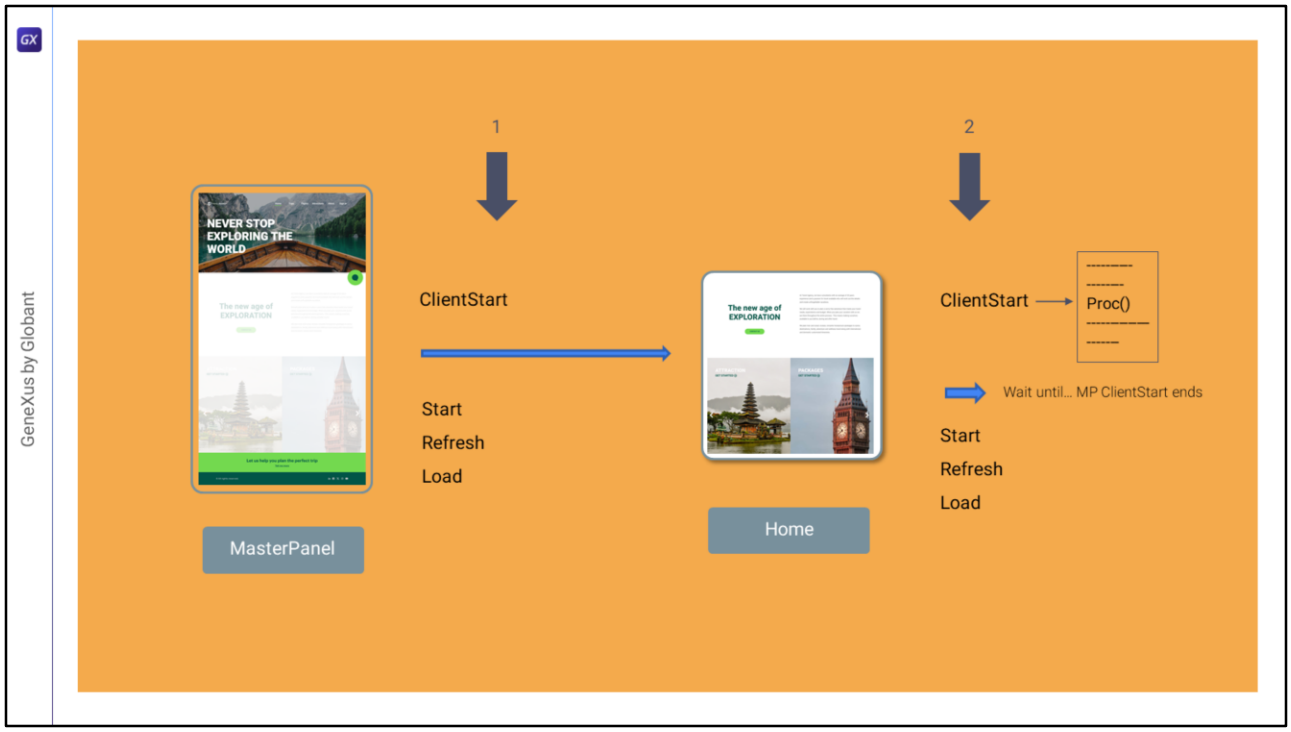
Decíamos, entonces, en este momento se toman en cuenta todas las propiedades del tiempo de diseño. En particular, entonces, se asignan las clases definidas estáticamente a nuestros controles.



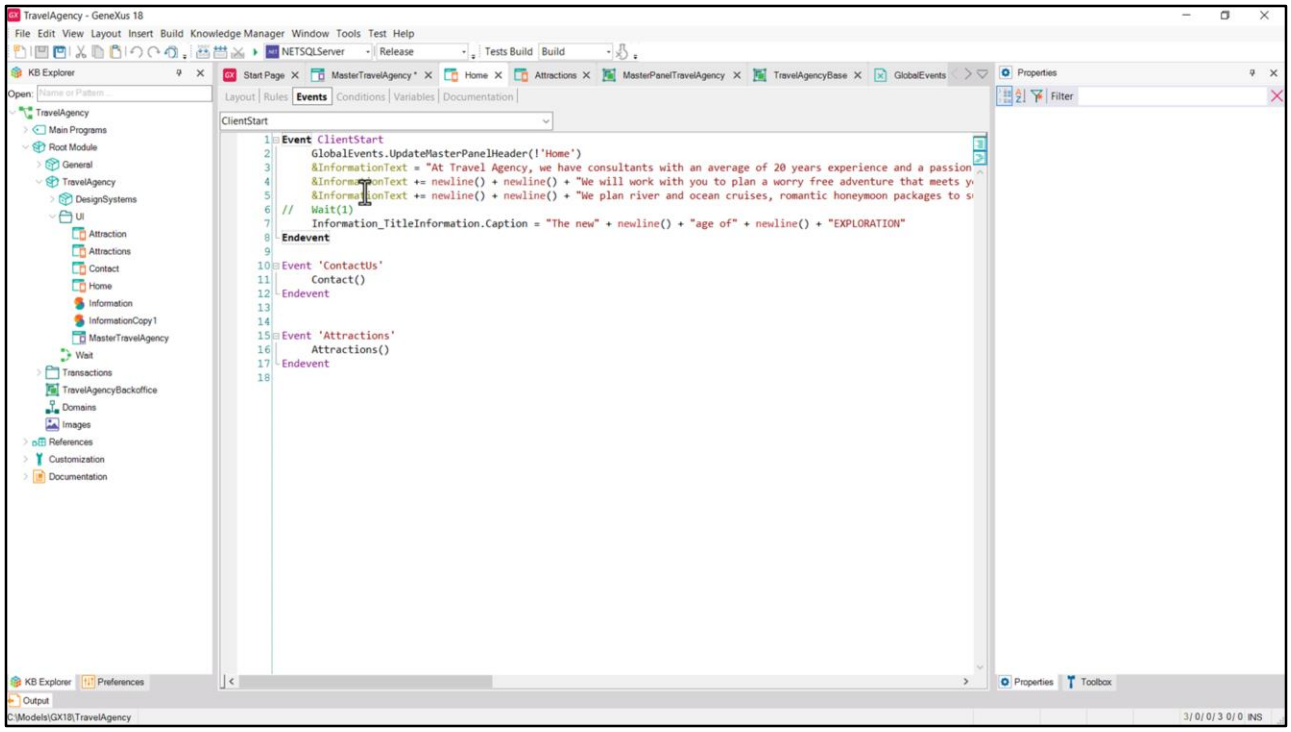
Bien, luego se ejecutará, decíamos, el evento ClientStart, que lo teníamos comentado… entonces allí no pasa nada.



Y, como consecuencia de todo esto, queda renderizado el Master Panel con lo que se tiene hasta el momento: las variables de imagen y título van a estar vacías, y el menú con la opción Home seleccionada.



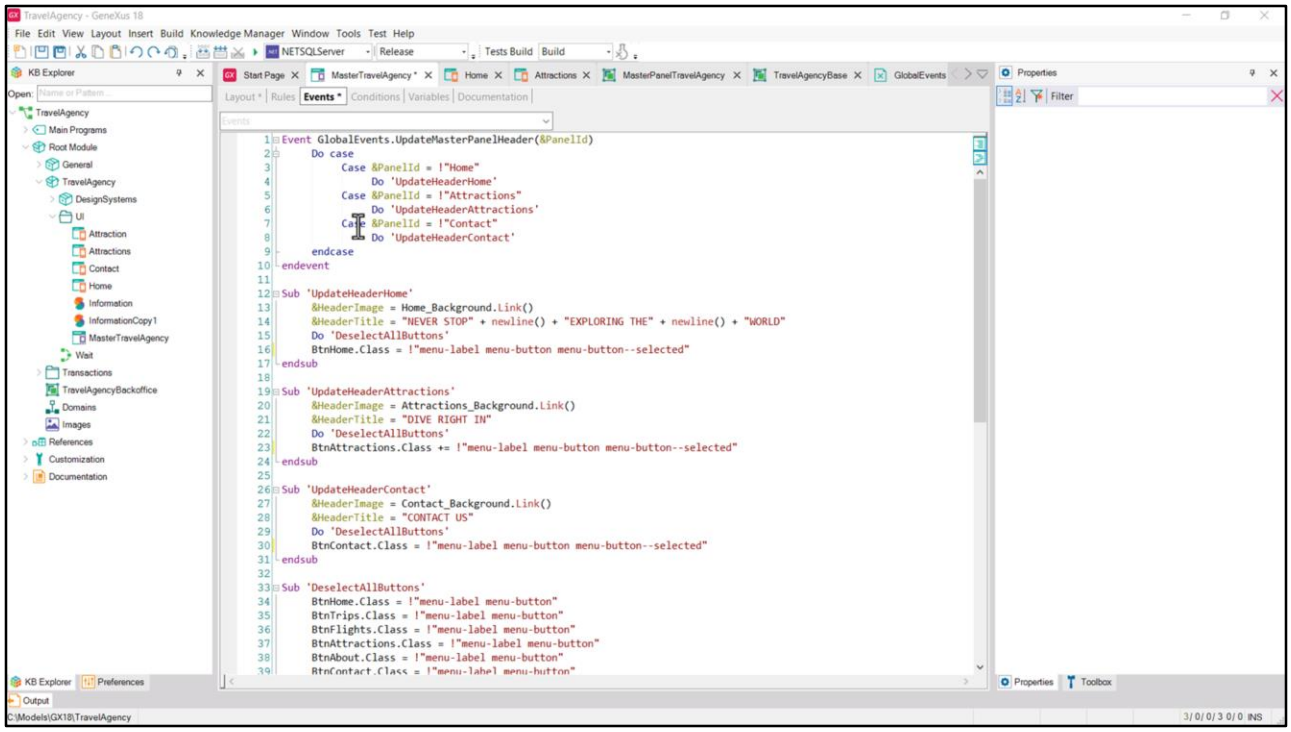
Luego se ejecuta el Panel propiamente dicho, y su evento ClientStart.



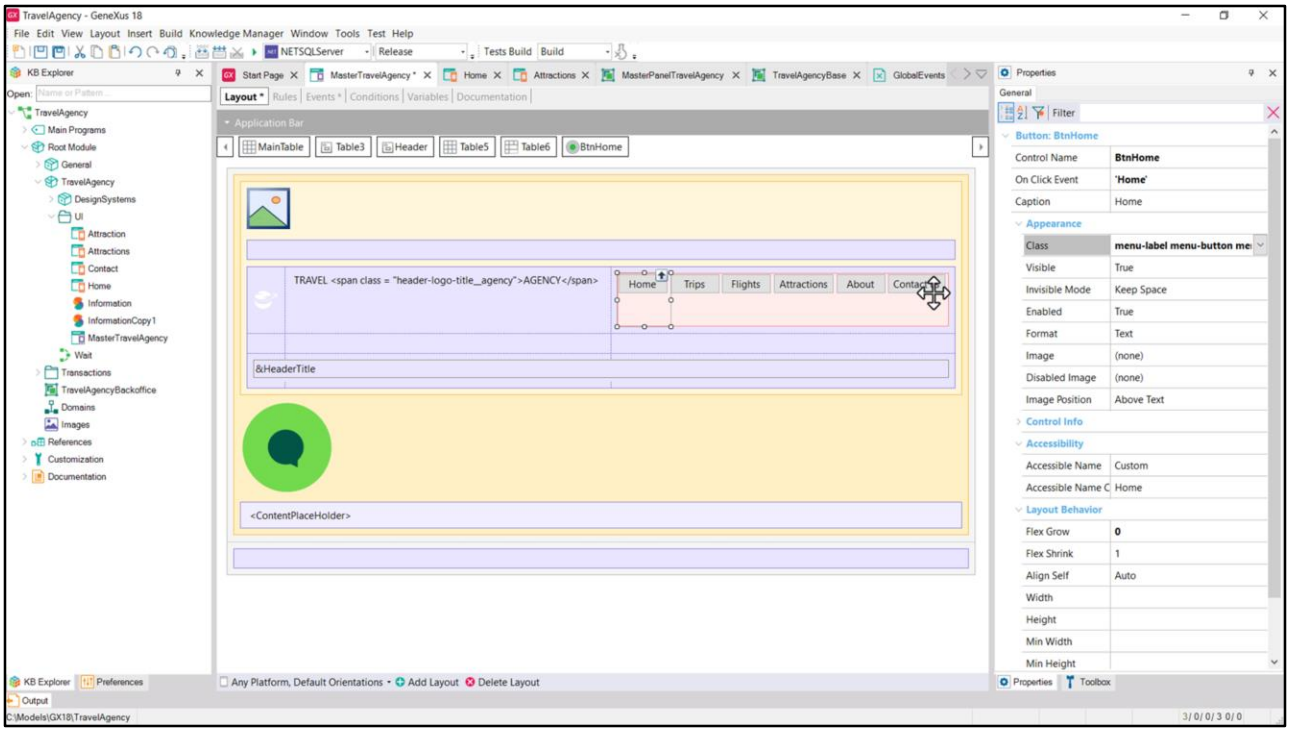
Entonces se va a ejecutar el evento ClientStart…

Y si lo observamos tiene no sólo la invocación al evento global, sino también estas asignaciones.

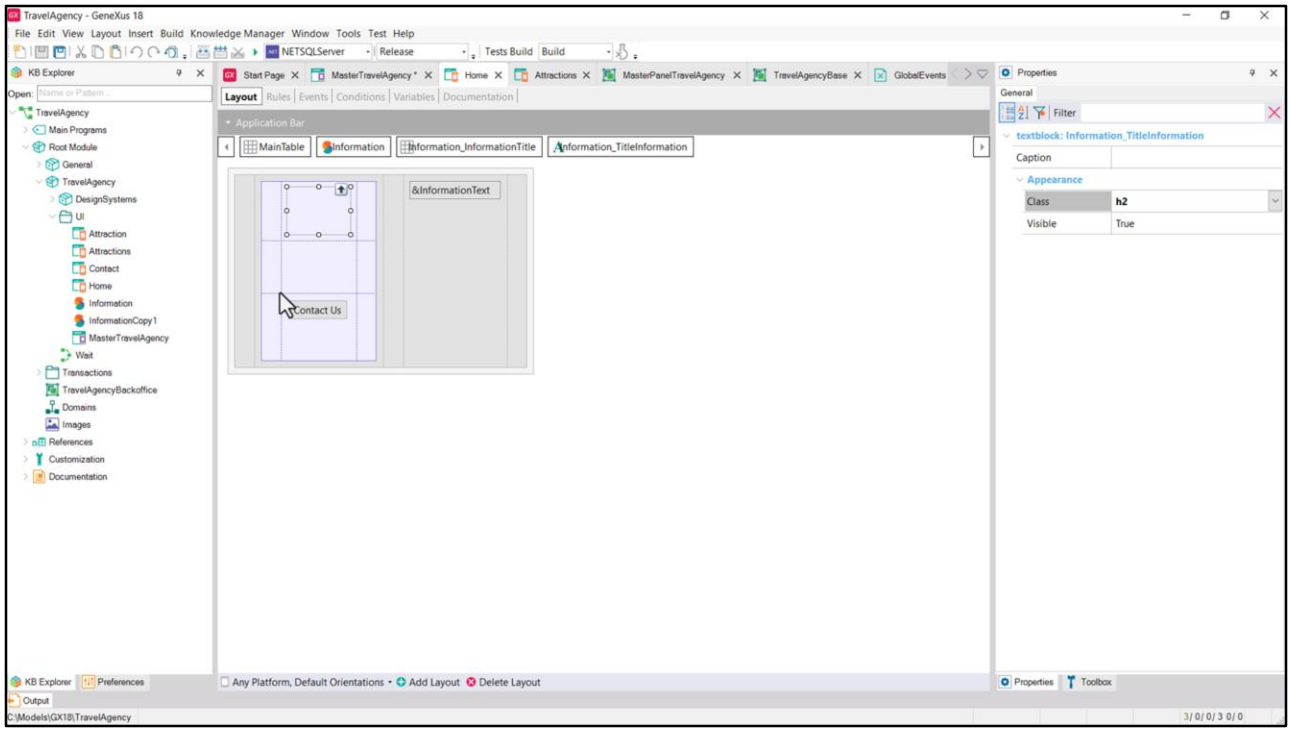
Tuve mis dudas en su momento de si colocar la invocación al evento global al principio o al final del código. El resultado va a ser el mismo, dado que lo que va a suceder es que va a disparar el evento global de manera asincrónica, pero sin bloquear la ejecución, es decir, no va a detenerse y esperar a que termine para proseguir con este siguiente comando, sino que va a continuar hasta terminar.



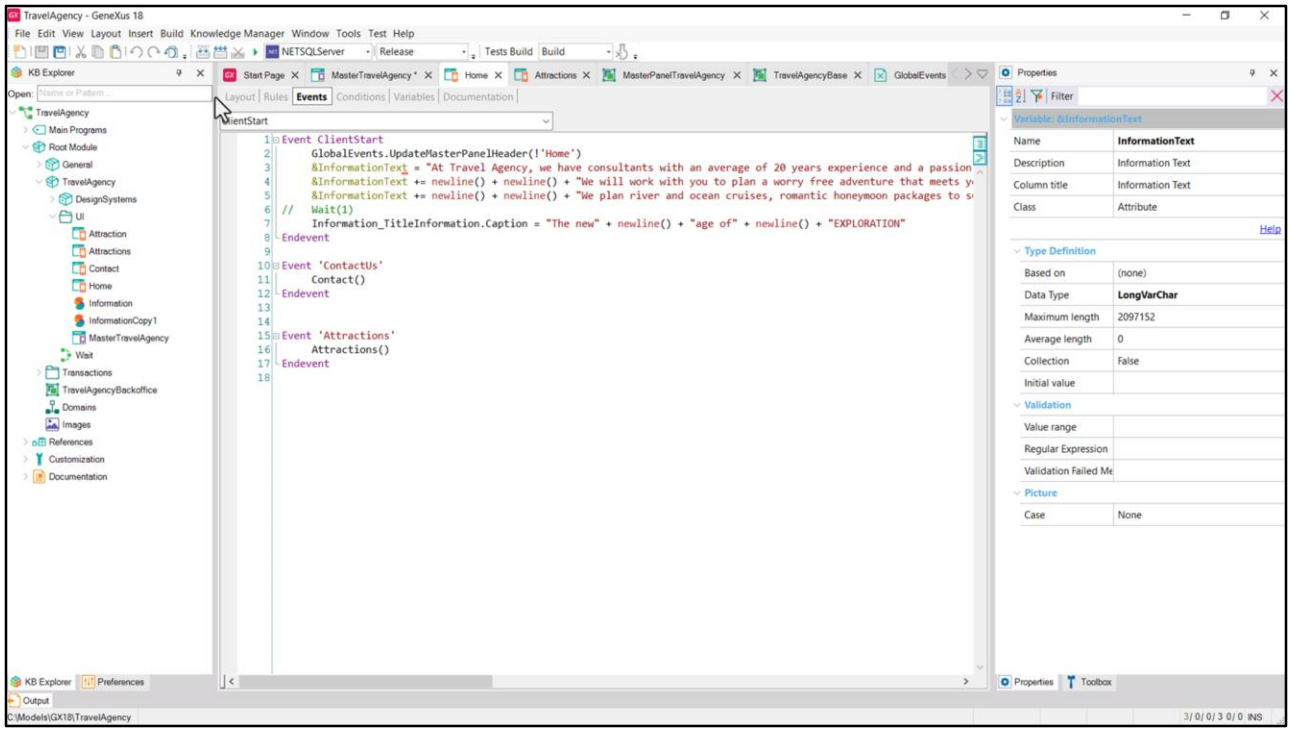
Como el evento global está siendo escuchado por el Master Panel, que está activo en el cliente en esta ejecución, entonces ejecutará su código.



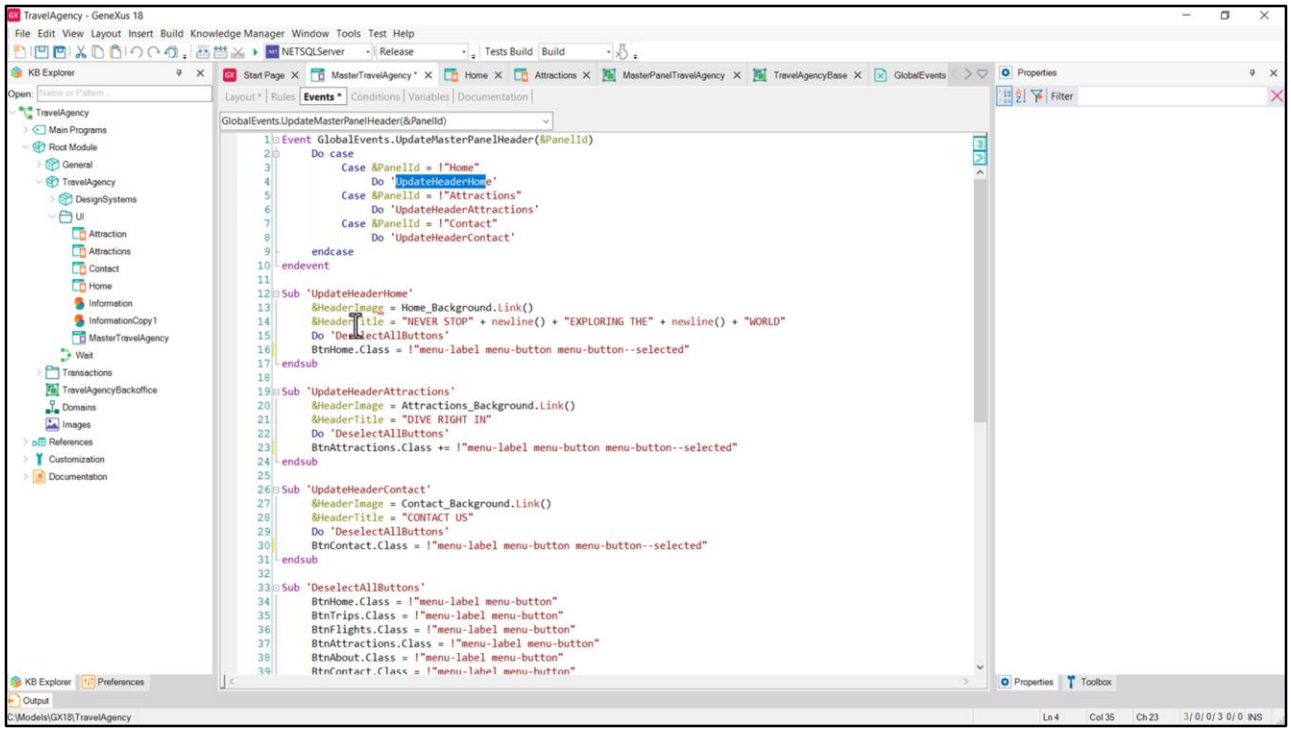
Así que resumiendo el estado de la ejecución hasta este momento: se renderizó la pantalla del Master Panel primero, con la imagen vacía y el texto vacío, pero las clases para los botones del menú aplicadas, así que el menú estaría con Home resaltado.



Inmediatamente se renderizaron estas partes del layout de Home…

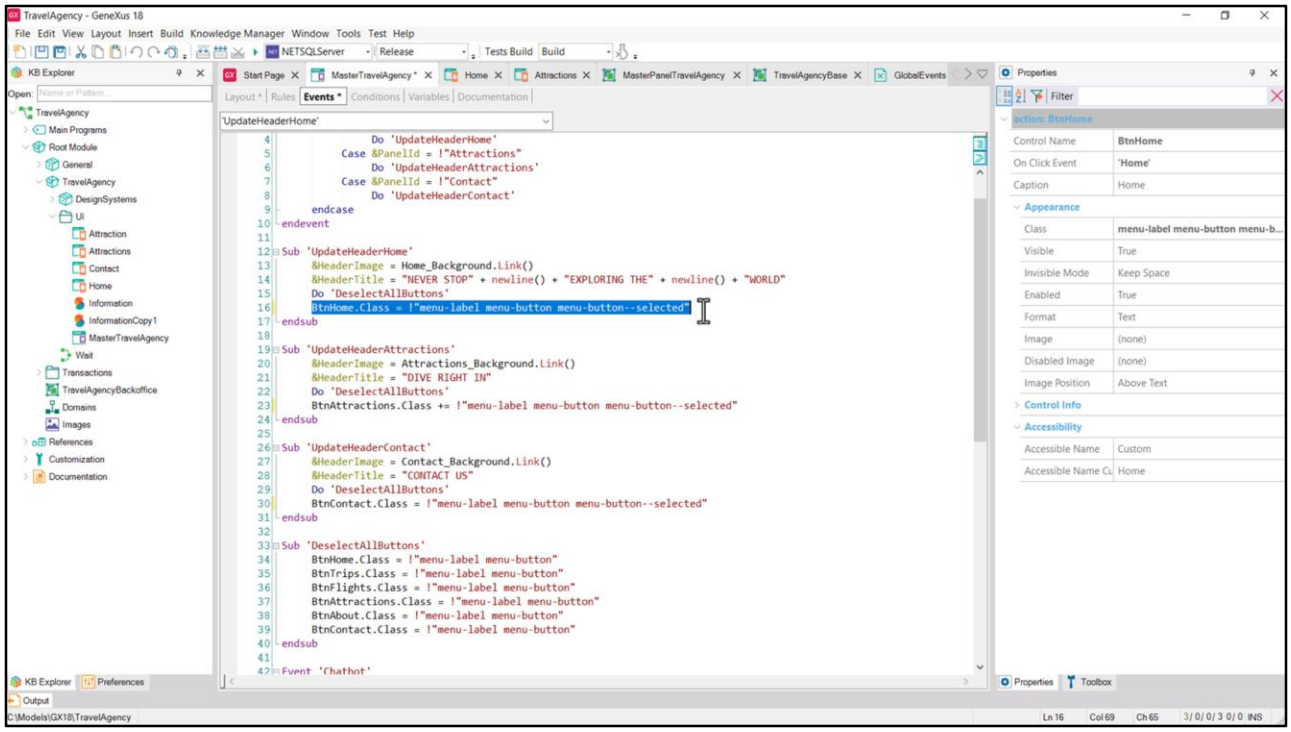


…y en este preciso momento…



…se está ejecutando el Do case del evento global, evento que siempre es del cliente.

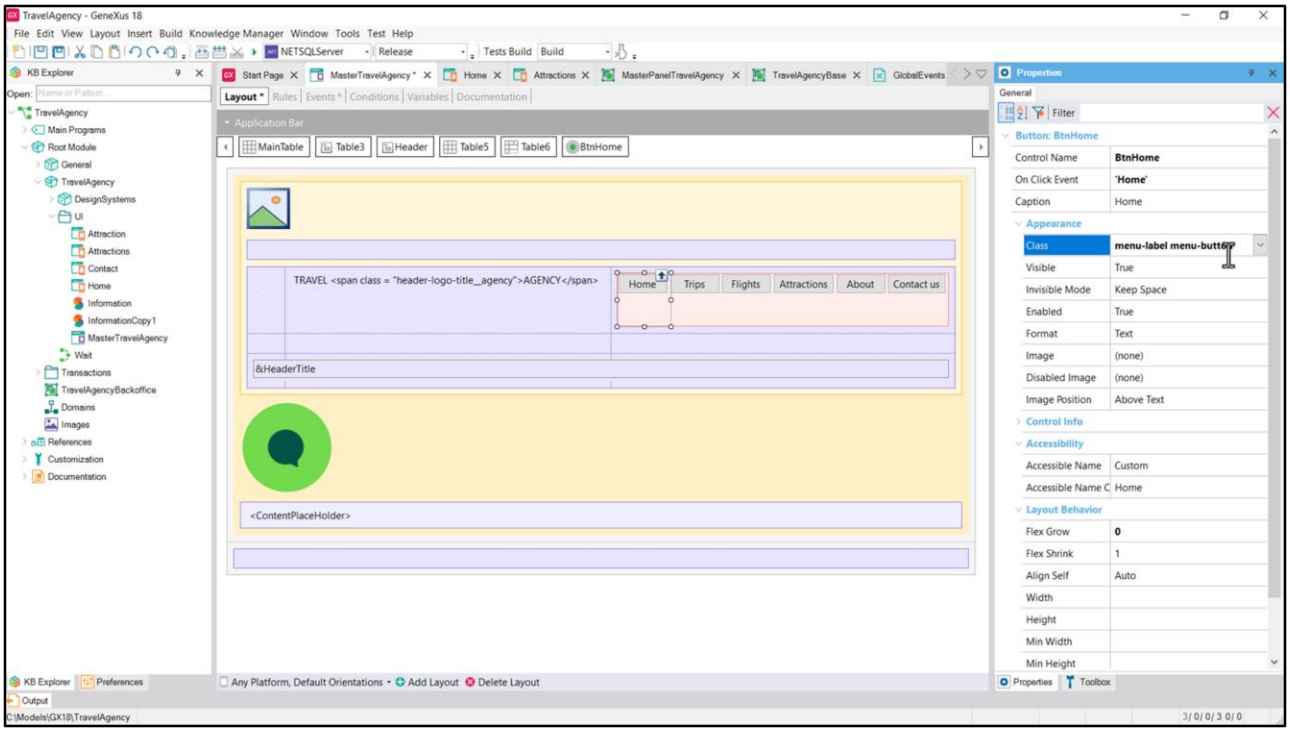
Entonces aquí se ejecuta esta subrutina… que asigna valor, ahora sí, a las variables &HeaderImage y &HeaderTitle.



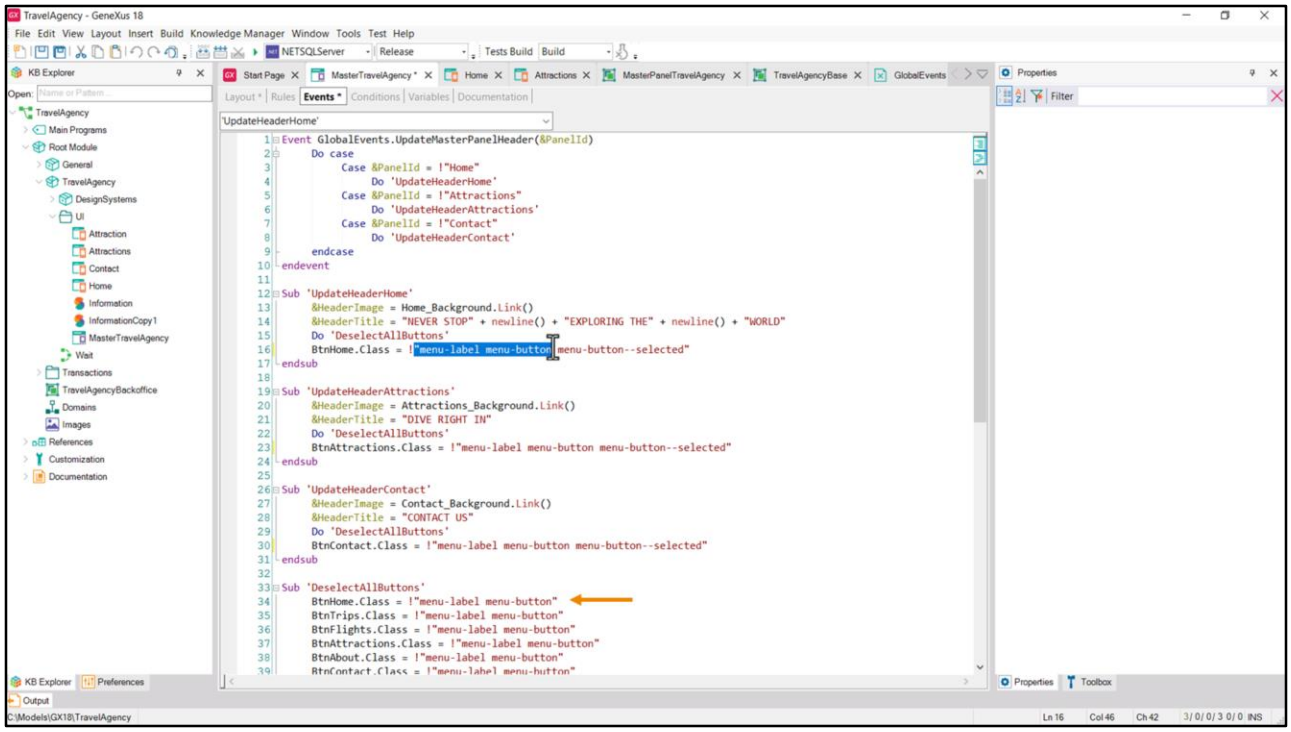
Y le sobrescribe en esta otra subrutina a todos los botones sus clases, por lo que la del BtnHome que tenía 3 clases ahora pasa a tener 2. Pero inmediatamente, dado que es el panel Home el que estamos ejecutando, vuelve a recuperar las mismas 3 clases.
Cuando todo esto termina, se vuelve a renderizar entonces esta parte del Header.

En total en este proceso hicimos 2 renderizaciones del botón Home. La primera con las clases estáticas, la segunda al disparar el evento global. Justo tal vez para el caso de Home pasa más disimulado, pero pensemos qué pasaría si estuviéramos cargando el panel de Attractions, no el de Home. En la primera renderización quedará como seleccionado el botón Home, y en la segunda, cuando se ejecute el evento global, allí el botón Home pasará a quedar deseleccionado y ahora pasará a estar seleccionado el de Attractions. Esto podría producir un pestañeo, entre la primera renderización y la segunda.

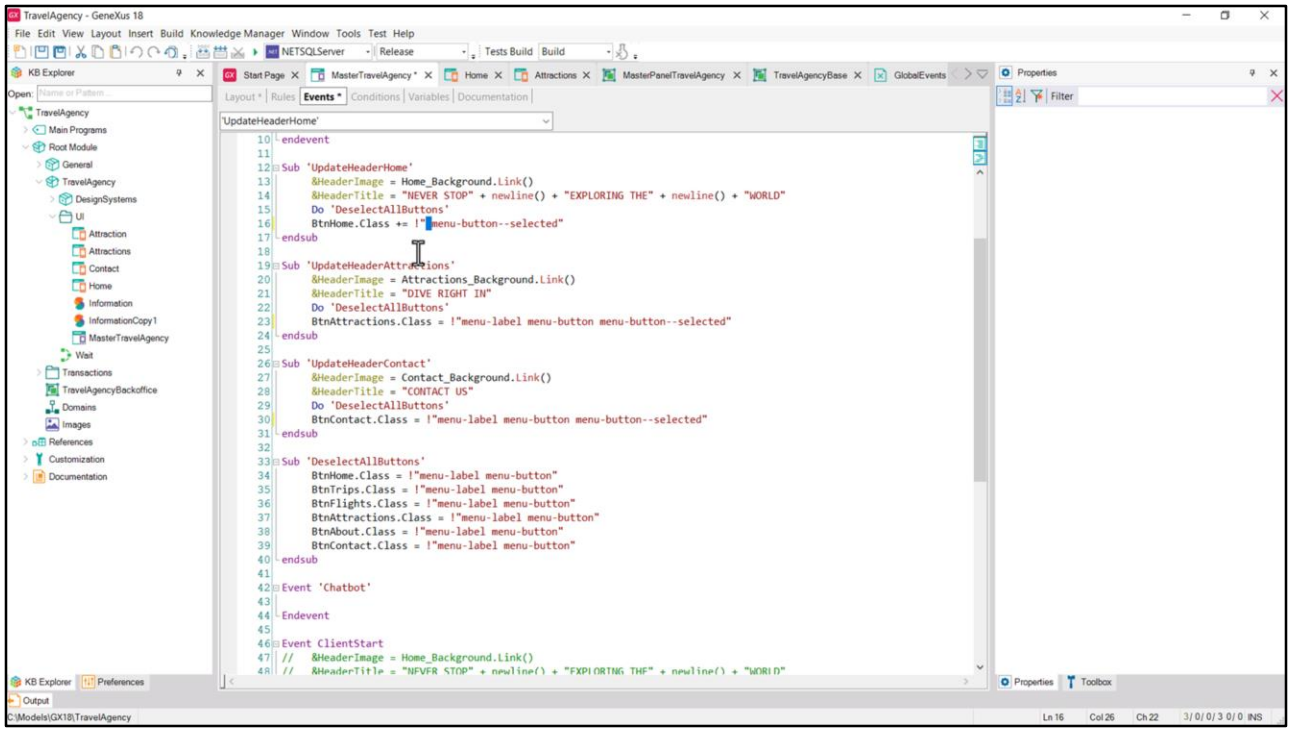
En verdad en este caso será un tanto inapreciable porque todo se resuelve en el cliente inmediatamente.



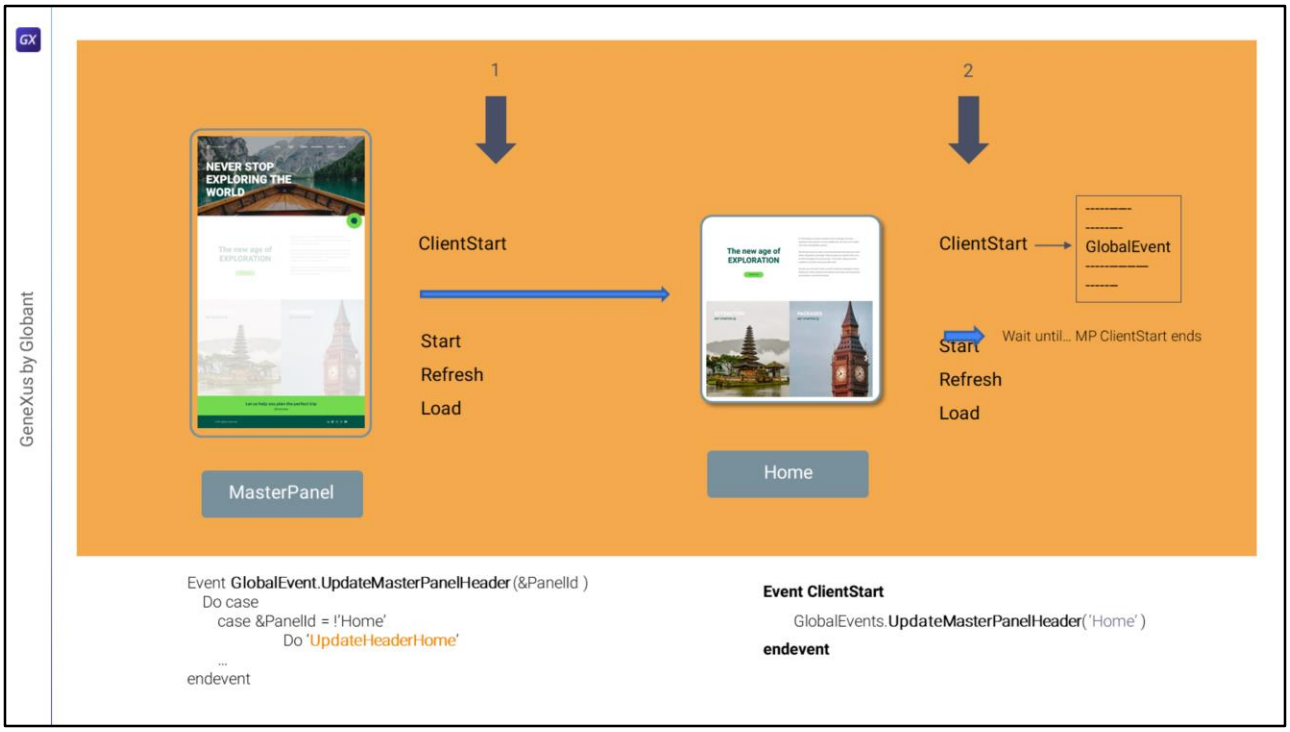
Pero para hacer las cosas bien, quitemos la asignación estática a la clase seleccionada del botón Home. Quedan así todos uniformes, con las clases default, por lo que cuando se sobrescriban con los mismos valores en la segunda renderización, no habrá chance de que se vea pestañeo alguno.



Llámenme obsesiva, pera me está molestando un poco esto… ¿por qué le vuelvo a decir que lleva estas dos clases, ¡si ya está dicho aquí!?



Vamos a utilizar mejor el operador “más igual” para que a lo que tenía, le agregue lo que sigue. Tengan cuidado de dejar un espacio en blanco aquí, porque esta es una concatenación de strings, y si no, me va a quedar pegado esto a esto otro. Hago el mismo cambio para los otros dos casos y ejecutamos.



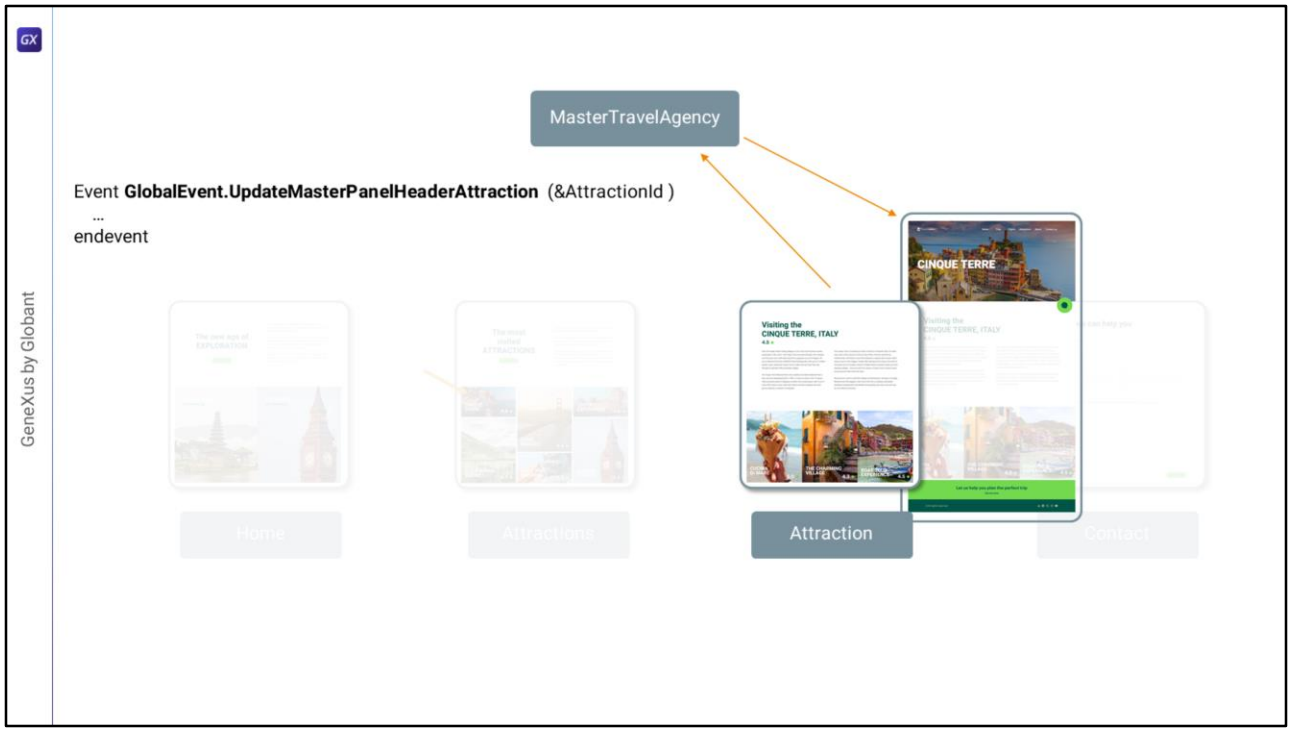
Resumiendo entonces:

Al ejecutar nuestro panel, se empieza por su Master Panel, considerando los controles del layout de acuerdo a sus propiedades estáticas. Luego se ejecuta el evento ClientStart, que en nuestro caso no tiene ningún comando asincrónico (de hecho no tiene nada programado) así que se renderiza el Master Panel con lo que se tiene hasta aquí.

Se da la orden de construir el panel para proyectarlo en el ContentPlaceHolder, lo que se hará en esta línea de ejecución concurrente, y paralelamente, en la primera, se ejecutan los eventos Start, Refresh y Load que están vacíos. Así que no sucede nada.

En la línea de ejecución del Panel, en su construcción, se consideran las propiedades estáticas de sus controles, y se ejecuta el evento ClientStart. En este caso tenemos el disparo de un evento global, que es escuchado por el Master Panel, así que se va a dar la ejecución concurrente de este evento y la continuación de la ejecución del ClientStart del Panel, que no se va a detener por la otra. Cuando termina se renderiza el panel y como el ClientStart del MasterPanel terminó hace rato (antes de lanzar la construcción, de hecho), entonces seguirá por la ejecución de Start, Refresh, Load.
Paralelamente el Master Panel al ejecutar el evento global, como modifica contenidos de la User Inteface, los va a renderizar.

Como ya les comenté, se está trabajando para que al navegar entre Panels con el mismo MasterPanel no se vuelva a construir el MasterPanel y así se mejore la performance y user experience.



Ahora tendríamos que ver lo que nos quedó pendiente: cómo hacer para que el Master Panel cargue en el Header imagen y título de la atracción que recibió por parámetro el panel Attraction al ejecutarse…

…y que lo recibió cuando el usuario seleccionó una atracción turística en este grid.

Sabemos que será a partir de este otro evento global. Para no extenderme más aquí, lo haremos en el próximo video. Los espero.


GeneXus by Globant
GeneXus™
by Globant
training.genexus.com