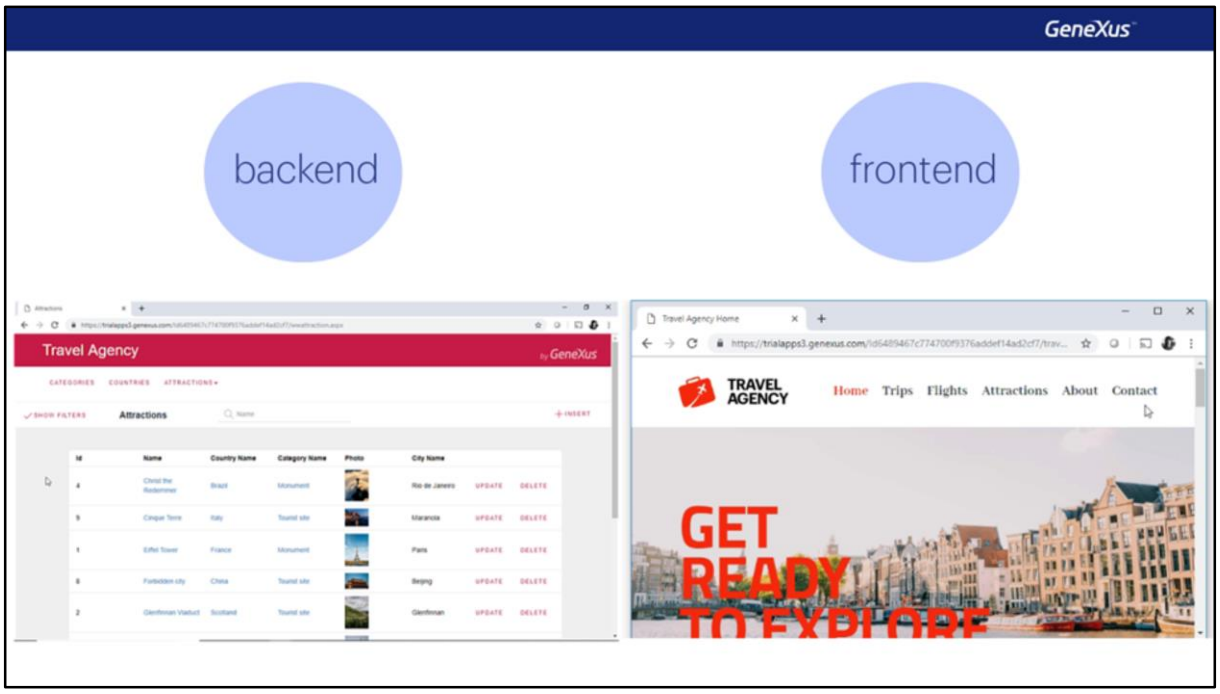
Design System en GeneXus

Default

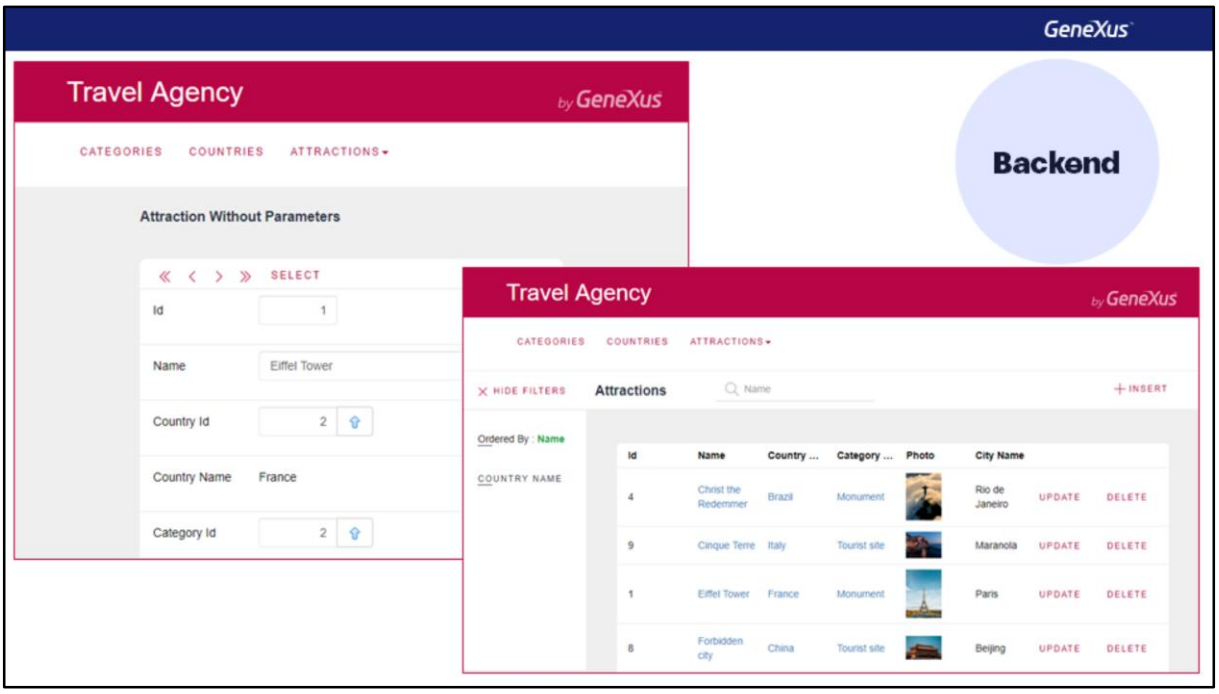
GeneXus 16



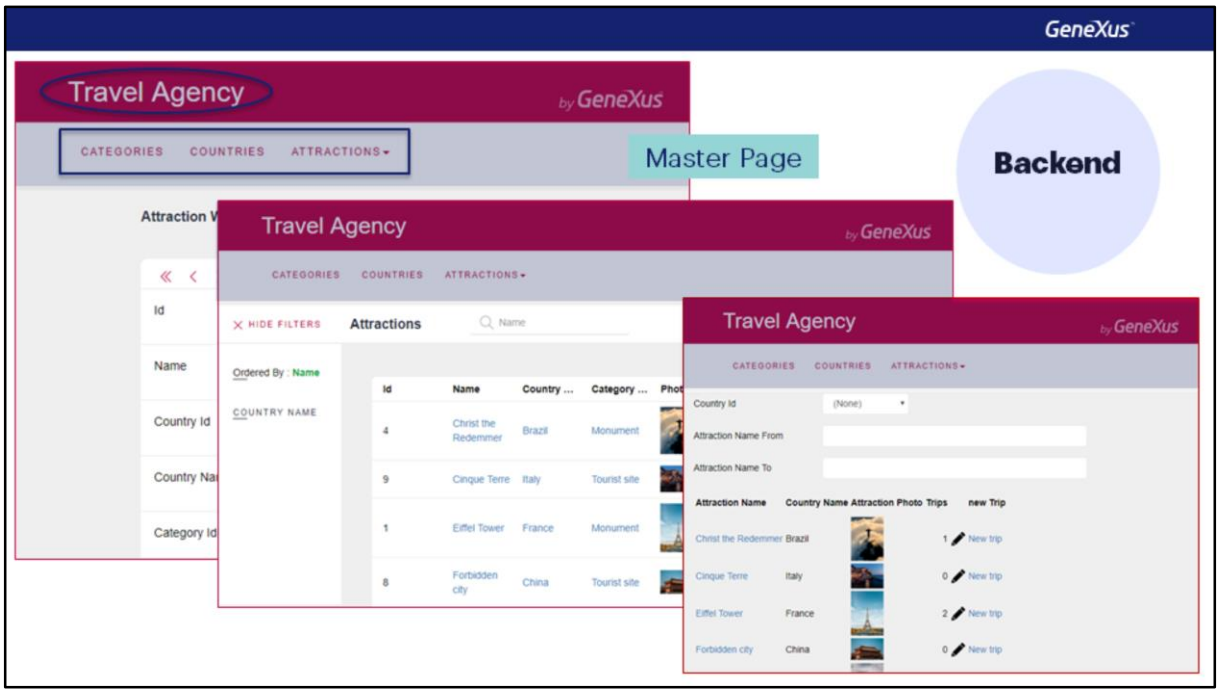
[DEMO: https://youtu.be/l_VA02JfGYQ]

Dijimos repetidamente que hasta ahora más bien habíamos estado trabajando en pantallas del **backend**. Sin embargo, como veremos sobre el final, el **frontend** es completamente análogo. La diferencia más grande es el mayor empeño que pondremos en su diseño.

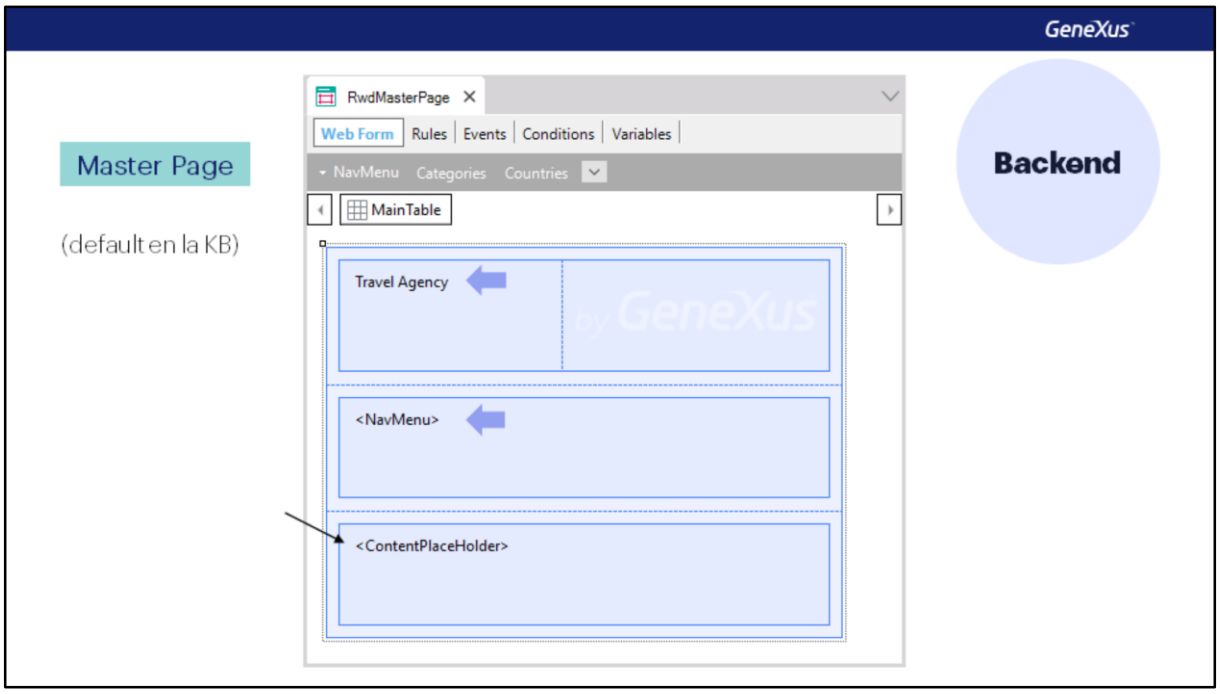
Empecemos por el **backend**...



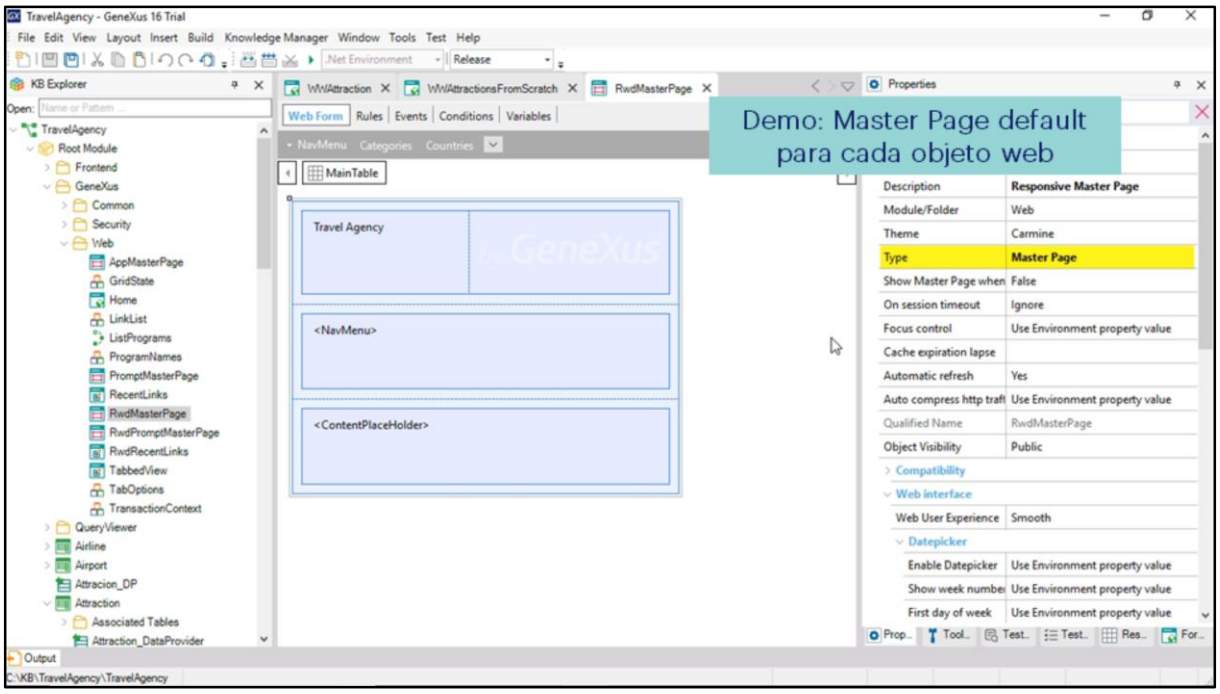
Si prestamos atención a las transacciones que venimos utilizando, y a los paneles creados por el patrón Work With, vemos que hay elementos que dan uniformidad y coherencia, sin que nos hayamos preocupado en lograr esto. Lo que significa que GeneXus ya está haciendo algo por nosotros en lo referido a brindarnos un Design System predeterminado, aunque sea elemental.



Por ejemplo, vemos que todas las pantallas (incluso las que hicimos de cero, como este web panel) tienen un encabezado común. En nuestro caso lo hemos personalizado, cambiándole el texto default por “Travel Agency” y quitando la sección “Recents” que venía predefinida (¡ah, y hemos agregado una barra de menú!). La forma de conseguirlo es con el objeto web Master Page (Aquí vemos el text block modificado y el menú agregado).



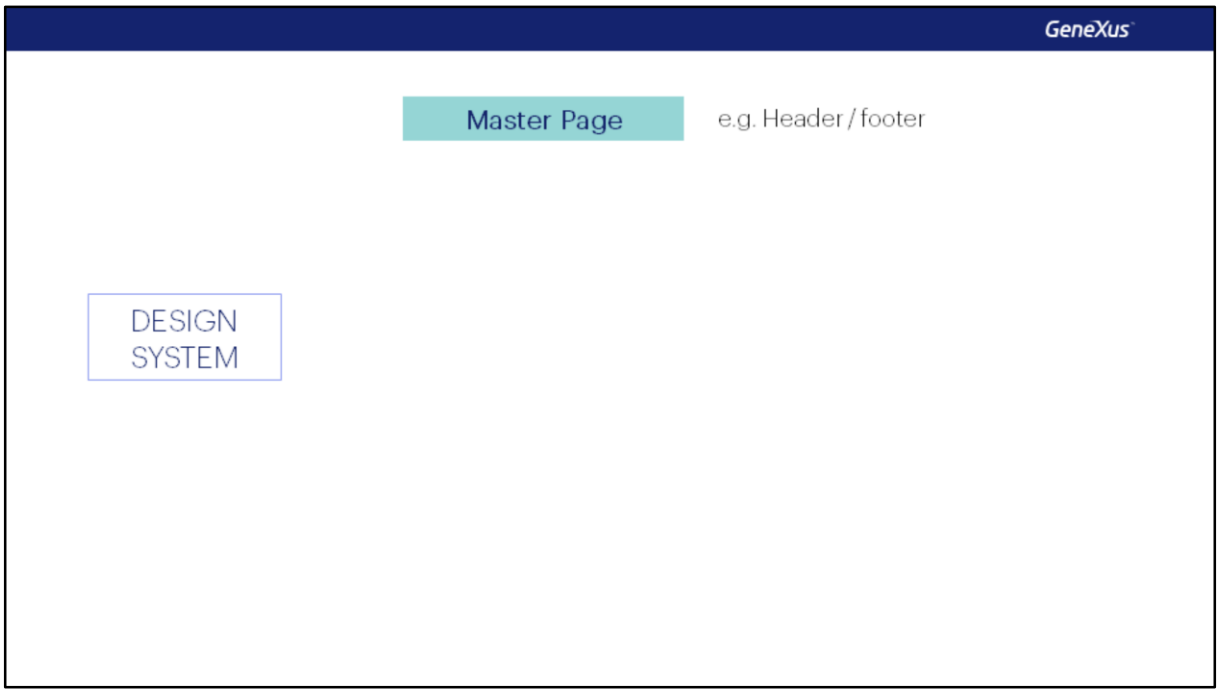
Por defecto, la versión de la KB viene con esta Master Page predefinida, lo que significa que se le aplicará a todos los objetos web que se creen.



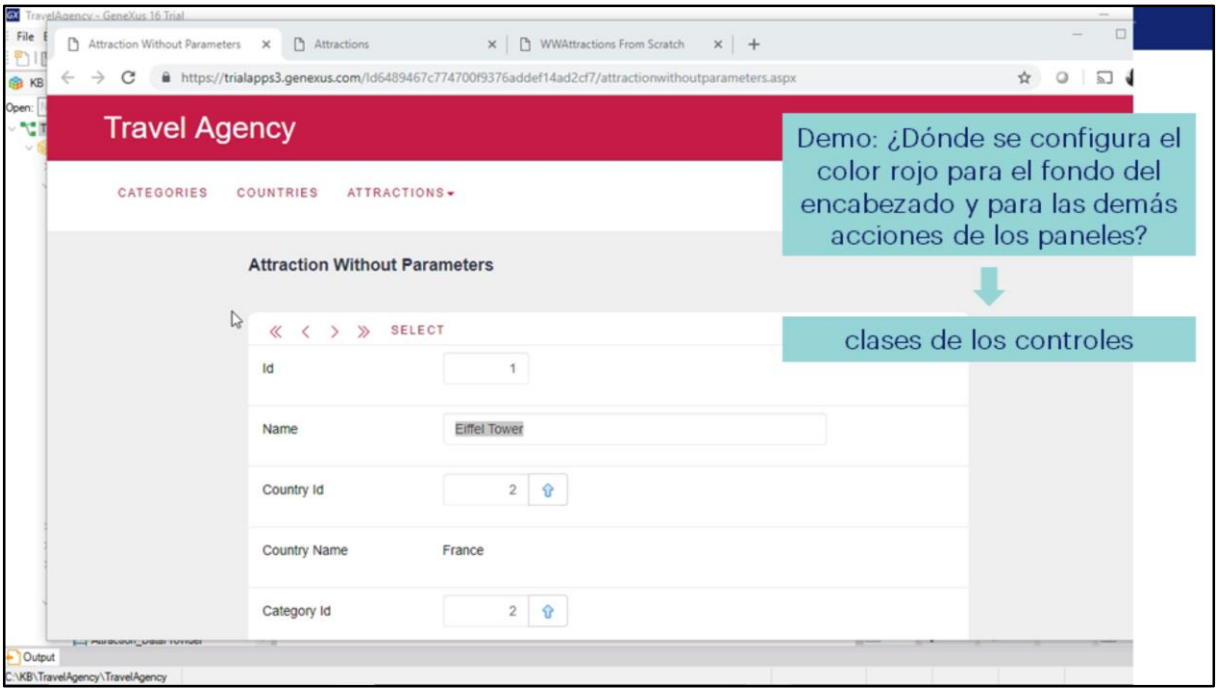
[Demo: https://youtu.be/PYBRdASQBvs]

Así, si observamos las propiedades de la transacción que veíamos en ejecución, vemos que tiene definida como su Master Page a esta, la default. Y lo mismo para el Work With y para nuestro panel.

Veamos que la Master Page contiene este control que indica que aquí es donde se cargará el contenido de cada página web.



Así que, por un lado GeneXus ya nos ofrece un objeto Master Page para centralizar lo que será la información común que queremos que las páginas compartan.

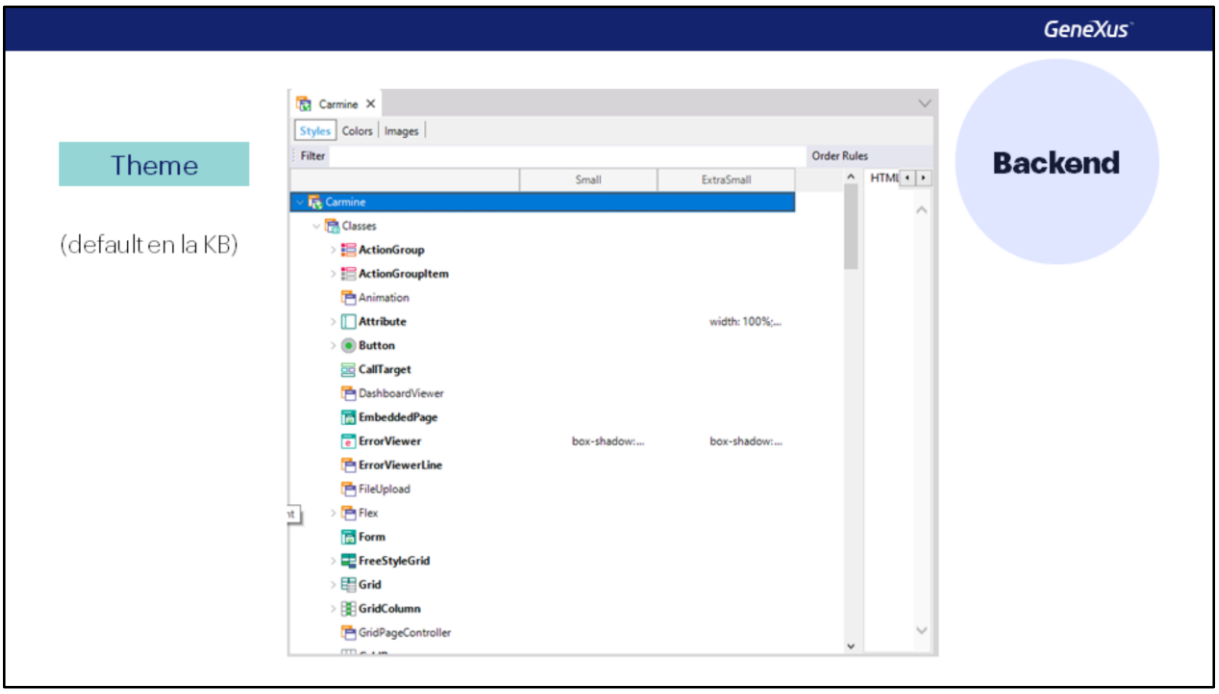


[DEMO: https://youtu.be/vkybJzp3uE0]

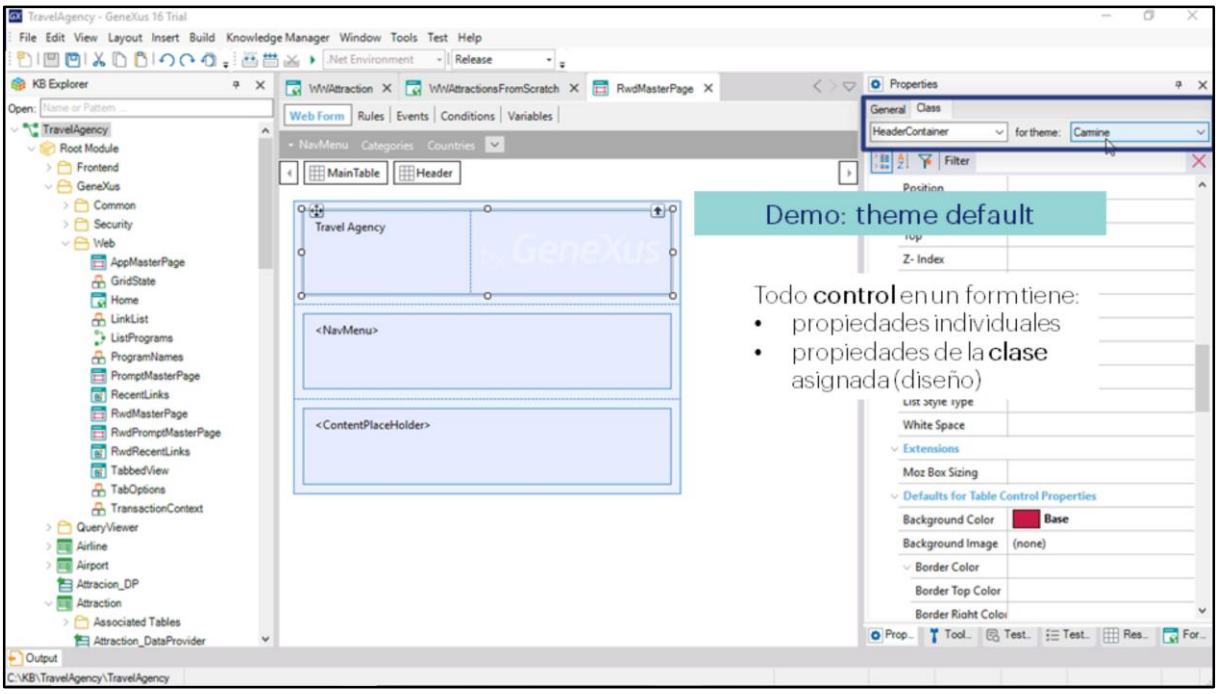
Por otro lado, podemos ver que el color de base, el que sobresale, es un tipo de rojo, que se utiliza no solo para el encabezado, sino para los botones y otras acciones que se le brindan al usuario. ¿Dónde se configura todo eso?

Sabemos que esta tabla tiene un fondo rojo en ejecución, pero ¡no lo estamos viendo! Sin embargo, observemos que el control (en este caso de tipo **tabla responsiva**) tiene un par de **clases** definidas, no casualmente bajo el grupo "Apariencia". Si vamos a la **solapa Class**, vemos que se están editando las propiedades de la segunda de las clases asociadas a esa tabla. Y vemos que allí están definidas cosas tales como márgenes , paddings, y ¡el color de fondo rojo que veíamos!

La gracia de las clases es que permiten centralizar el diseño de ese tipo de control (en este caso, tabla), y que, entonces, en nuestro caso otras tablas que se encuentren en el mismo o en otros forms puedan compartir la misma definición, lo que hace que si queremos cambiar por ejemplo el color de fondo de rojo a azul para esos controles, no haya que hacerlo uno por uno en cada control, es decir, en cada tabla, sino que directamente se hace en la clase y eso ya repercute en todos los controles que la tengan asociada.



¿Dónde se encuentran todas las clases definidas? ¡En el objeto Theme, Tema! Vemos que, así como había una Master Page predefinida, también hay un tema predefinido, de nombre Carmine.



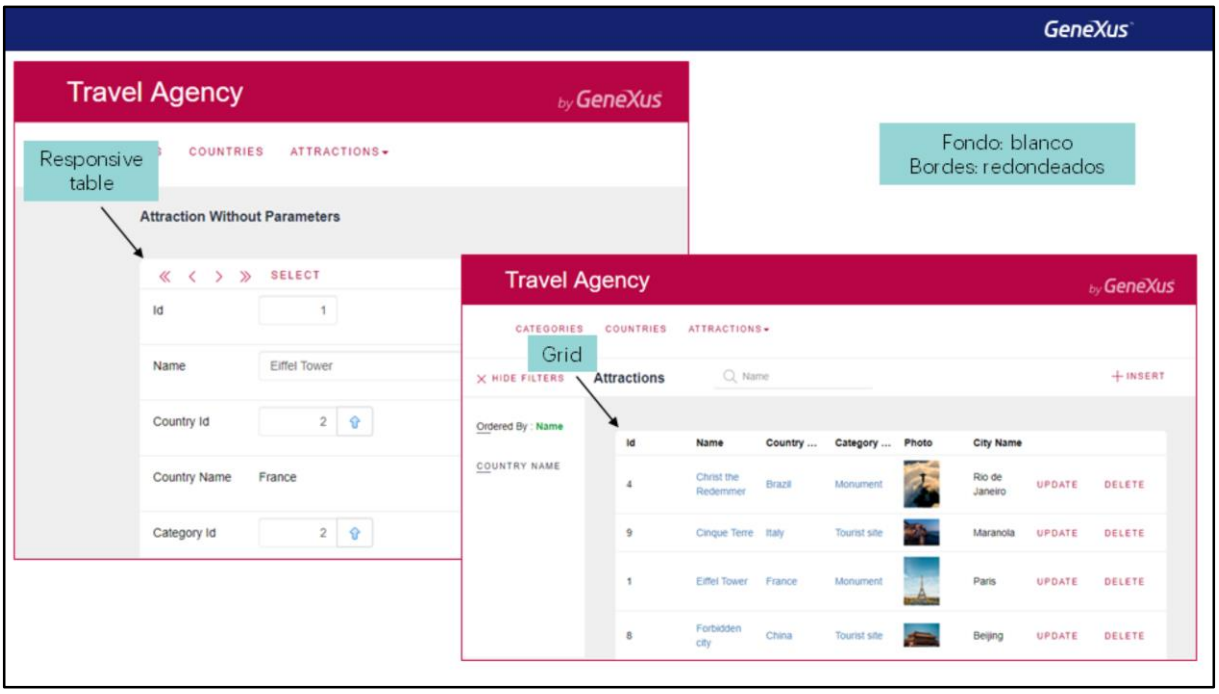
[DEMO: https://youtu.be/tkdOB-JPSgg]

Como veremos luego, hemos creado uno nuestro que será el que utilizaremos para el frontend. En nuestro backend usamos el default. Vemos que al abrirlo aparecen clases agrupadas por tipo de control. Aquí es donde está centralizado todo. Y además podemos ver en la solapa Colors, que muestra la paleta de colores del mismo nombre... Que es donde está definido el color Base.

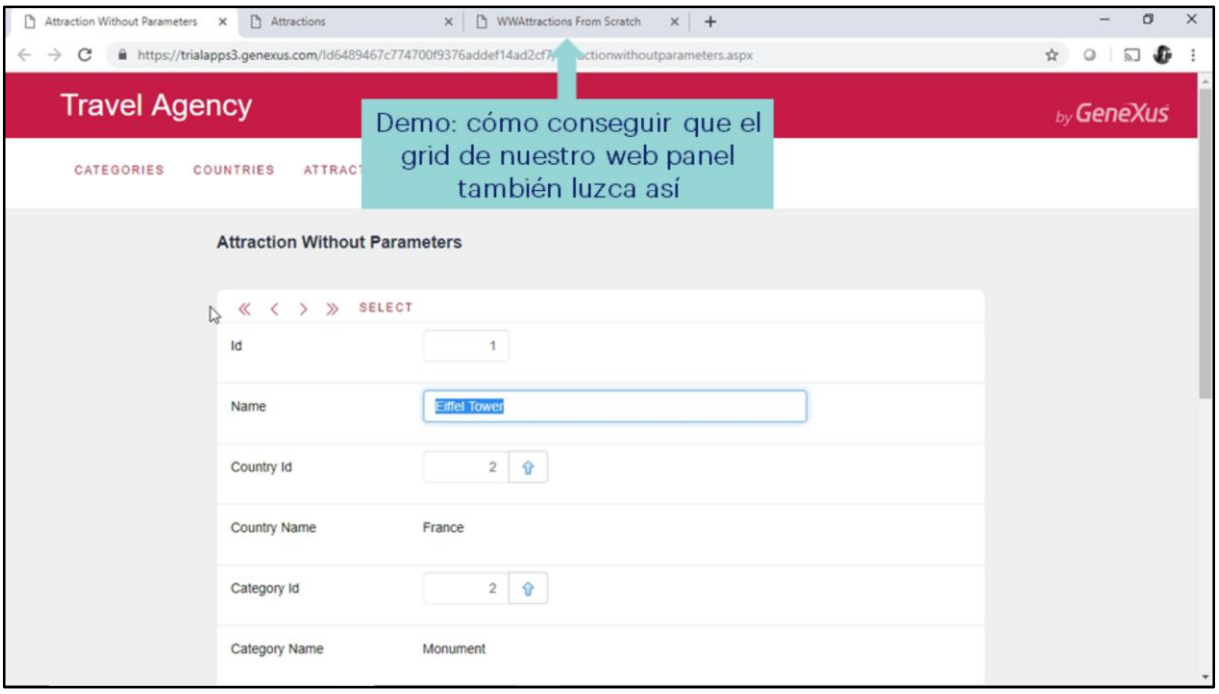
Si vamos a ver las opciones de Update o Delete del pattern Work with, que sabemos tienen el color base en ejecución, vemos que el tipo de control es Attribute/Variable. Y que tiene asignado como clase a esta, predefinida. Podemos aquí mismo editar sus propiedades sabiendo que lo que modifiquemos aquí se modificará a nivel central, en el objeto Theme, por lo que cualquier otro control atributo/variable que tenga esta como clase, se verá modificado.

En definitiva, todo control en un form tiene propiedades individuales, que se pueden configurar para él solito... Y propiedades que vendrán de la clase que tenga asignada, que son las que involucran más que nada los aspectos de diseño.

Ya observemos que en este caso particular en que el control es, además, una columna de un grid, aparece otra clase para asociarle, que es la clase que gobierna el diseño y comportamiento de la columna.



Y si observamos nuevamente el diseño predefinido de la transacción y el Work With, vemos que los datos más relevantes se muestran en blanco sobre gris, con bordes redondeados.



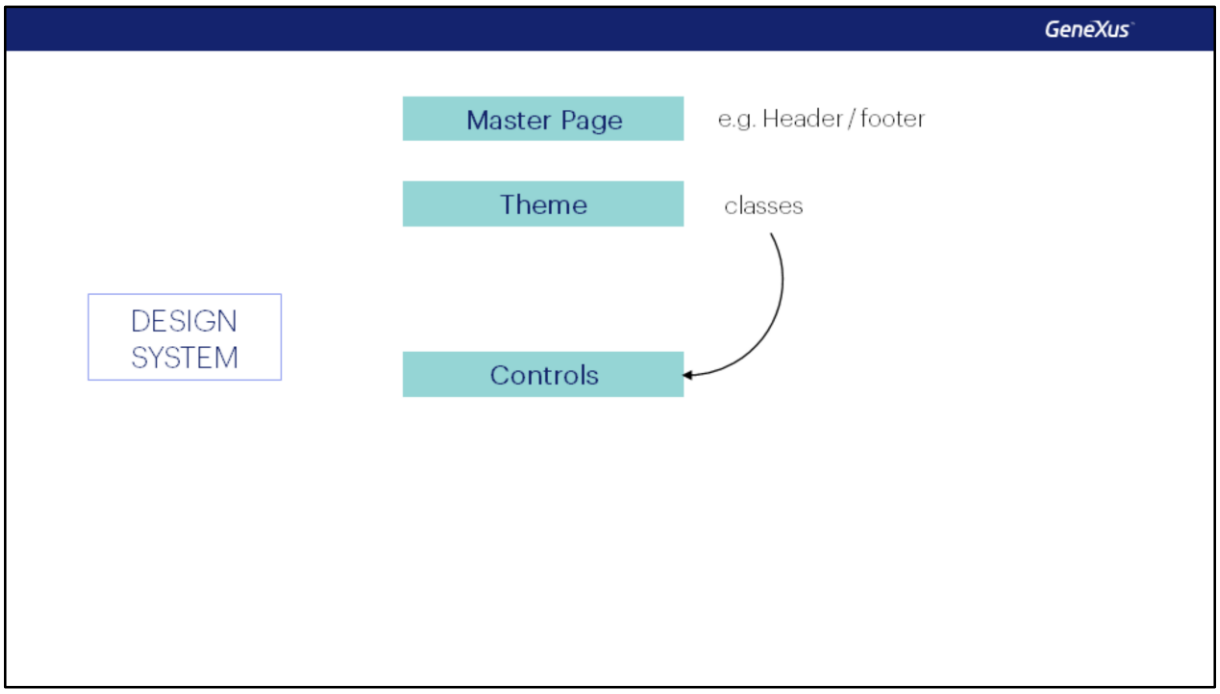
[DEMO: https://youtu.be/XhDqvve5E74]

Esto es así porque a la tabla donde se encuentran esos datos en la transacción se le asignó esta clase... que especifica como color de fondo el blanco y un radio para los cuatro bordes... ...y al grid del Work With esta otra clase, que especifica también blanco como color de fondo, y los mismos valores que la otra para el radio.

En cambio, si vamos a ver el grid del web panel que nosotros creamos, vemos que tiene como clase la de nombre Grid, que no tiene nada de eso configurado... Y así se ve en ejecución, con todas las columnas pegadas, con el fondo default, gris. Observemos qué sucede si le cambiamos su clase por la misma que tiene el grid del WorkWith...

Y para mantener la uniformidad del Design System, esta acción del grid debería lucir como estas otras...

Y aquí no debería estar el título de la columna...



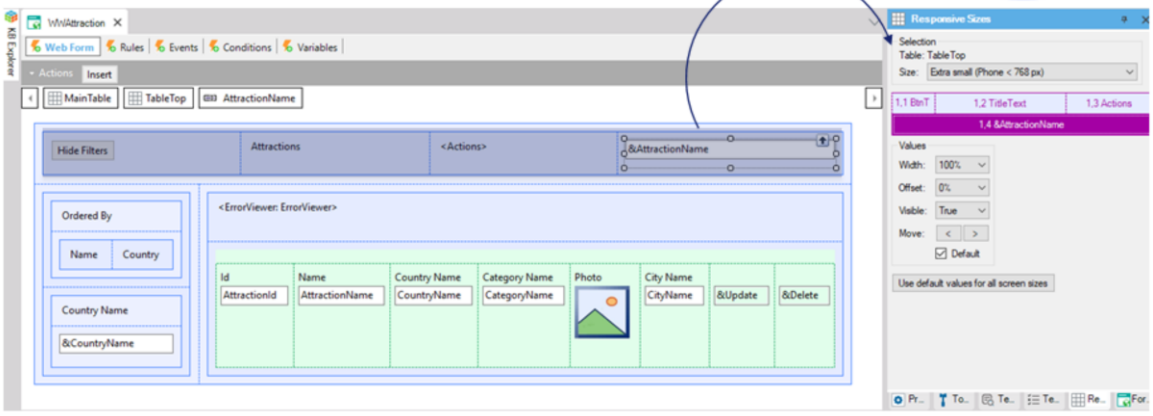
Podemos deducir de lo que hemos visto que un porcentaje importantísimo del Design System se especifica en el objeto Theme, a través de sus clases (las predefinidas y las que agreguemos). Estas clases son las que se asignan a los controles del form, permitiendo así abstraer el diseño, establecer una lógica, y de esta manera uniformizarlo de acuerdo a la función que asume cada control en el todo.

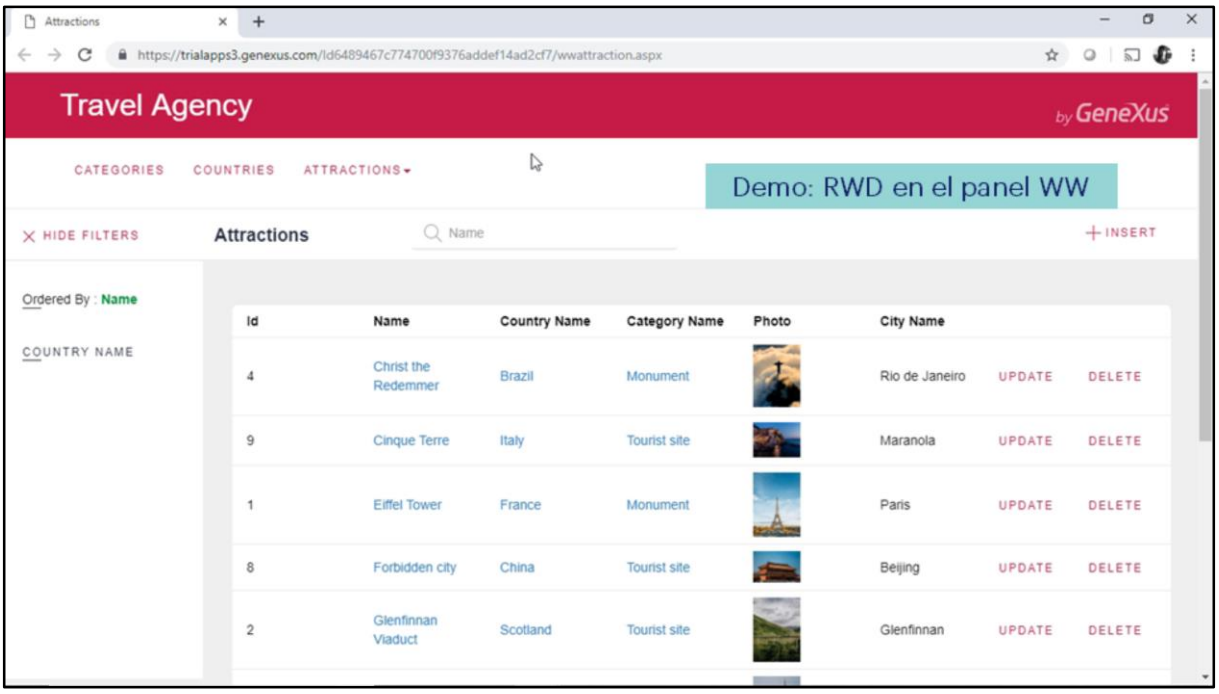
Por lo que una parte fundamental del diseño será identificar qué clases necesitaremos para los controles, y aplicárselas.

Responsive Web Design

Backend

- Responsive Sizes de las Responsive tables





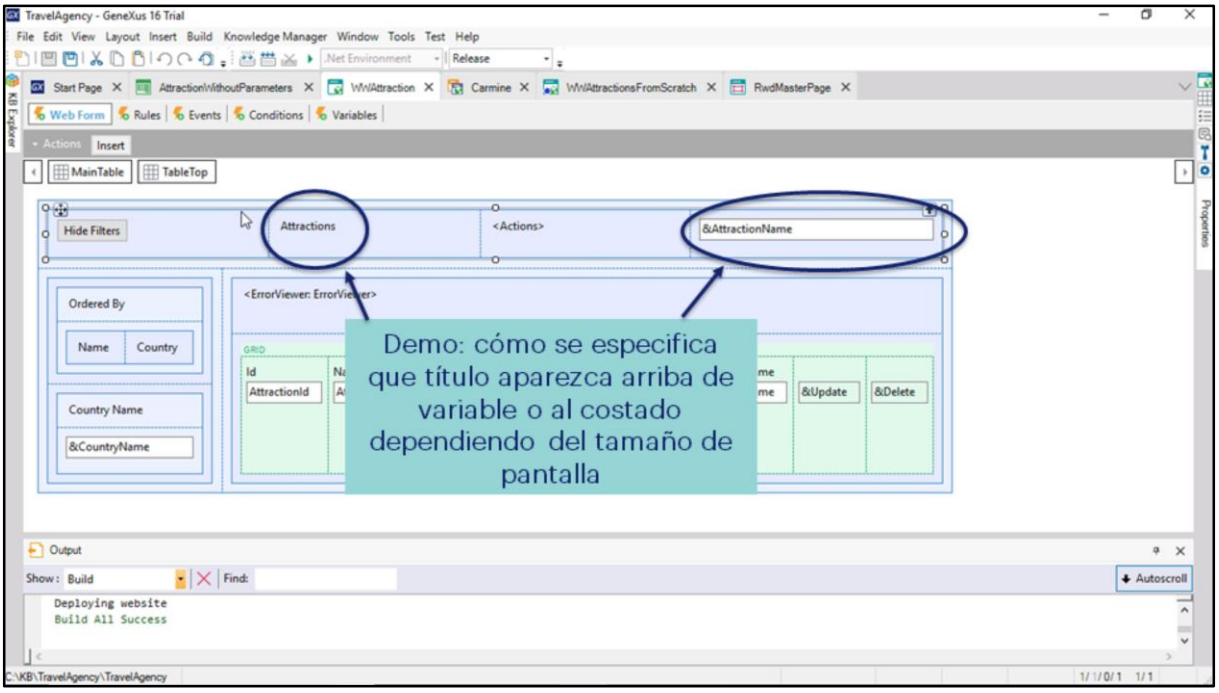
[DEMO: https://youtu.be/aRtwIstT7BU]

Volviendo a lo predefinido (recordemos que algunos navegadores nos ofrecen un modo de visualización –en este caso presionando F12– que nos permite variar el tamaño de la pantalla para apreciar cómo se vería)...

Podemos ver que si achicamos la pantalla del Work with de atracciones lo suficiente como para que corresponda a la pantalla de un teléfono, los filtros aparecen de otra manera, y el menú que habíamos agregado aparece como el típico menú de hamburguesa. Y, por ejemplo el campo para que el usuario pueda ingresar un filtro por nombre de atracción aparece abajo y no al costado.
Y con la disminución del ancho de pantalla, también desaparecen casi todas las columnas a excepción de la que muestra el nombre y las acciones de Update y Delete.

Habíamos nombrado a este comportamiento como “responsivo”, y al diseño web que lo considera como Responsive Web Design. Hoy en día no es aceptable un diseño que no sea responsivo.

Una parte de la responsividad se establece a través de las tablas responsivas...



[DEMO: https://youtu.be/qsbLGSHDzHM]

Por ejemplo, esta tabla, que vemos que es responsiva, contiene estos cuatro controles. El segundo corresponde a este textblock, y el último corresponde a la variable de filtro que veíamos. Si editamos las propiedades de la tabla, vemos ésta, "Responsive Sizes" (tamaños responsivos). Aquí se establece cómo se quiere visualizar el contenido de la tabla de acuerdo a cuatro tamaños de pantalla: **Extra small**, que corresponde a teléfonos, usualmente de ancho de pantalla menor a 768 pixeles; **Small**, que corresponde a tablets de ancho mayor o igual a 768 píxeles; **Medium** que corresponde a monitores de notebooks o Pcs mayores a 992 píxeles, y menores a 1200 píxeles; porque para los mayores o iguales está el tamaño **Large**.

Aquí podemos ver que si el tamaño es **Extra small**, la variable aparecerá debajo del título; mientras que si el tamaño es **Small**, la variable aparecerá a su derecha.

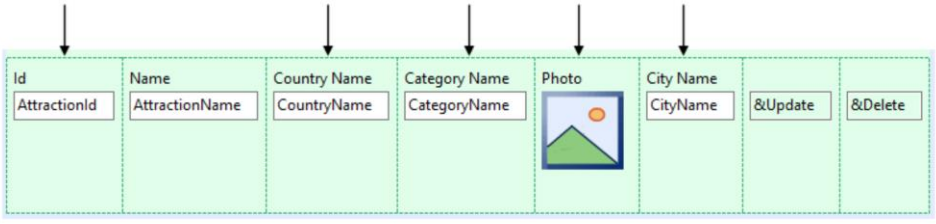
Decíamos que una parte importante de la responsividad, la más gruesa, se consigue haciendo uso de estas tablas y estas propiedades.
Pero otra parte, un poco más fina, se consigue a través de las **clases**.

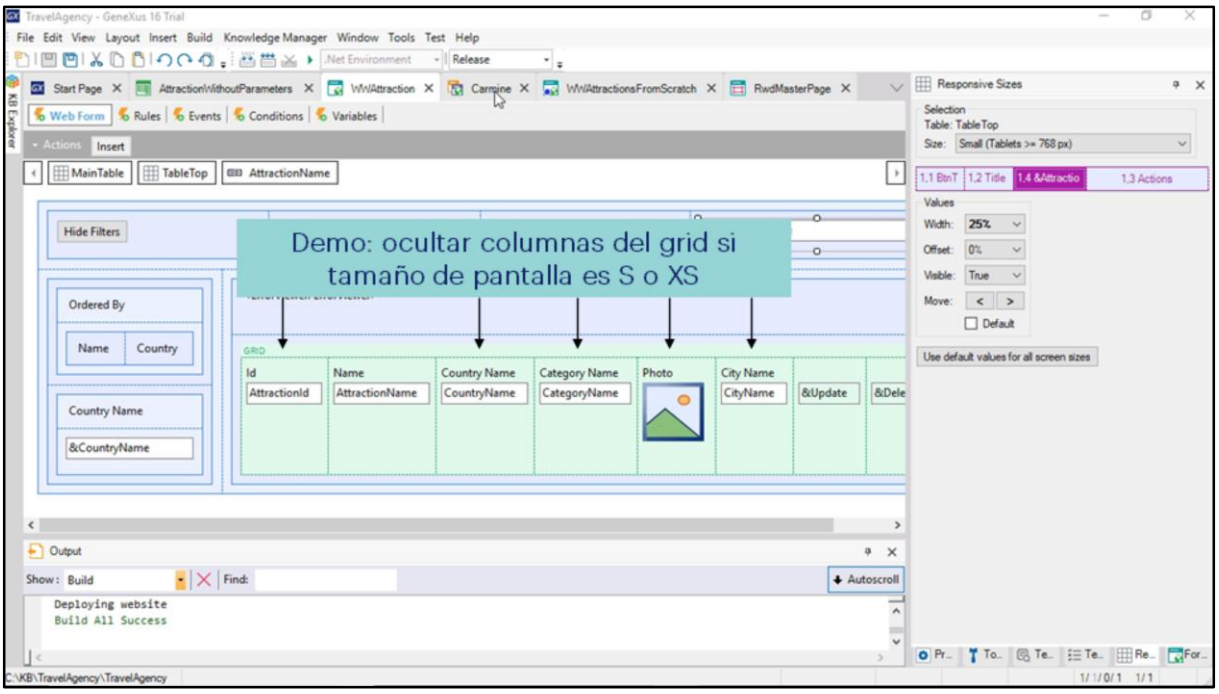
Responsive Web Design

Backend

- Responsive Sizes de las Responsive tables
- Rules del Theme: Small, Extra-small y Default

Ej: ocultar columnas del grid si tamaño Extra small o Small. Para tamaños mayores no.





[DEMO: https://youtu.be/WXz42v6mpNI]

Por ejemplo, el ocultar columnas del grid dependiendo del tamaño de pantalla.

Vemos que la clase de la columna del nombre de atracción es WWColumn, pero que a las otras columnas se les agrega, además, la clase WWOptionalColumn.

Si vamos al theme, vemos están apareciendo estas dos columnas que lo que hacen es permitirnos variar los valores de las propiedades de la clase, de acuerdo al tamaño de pantalla.

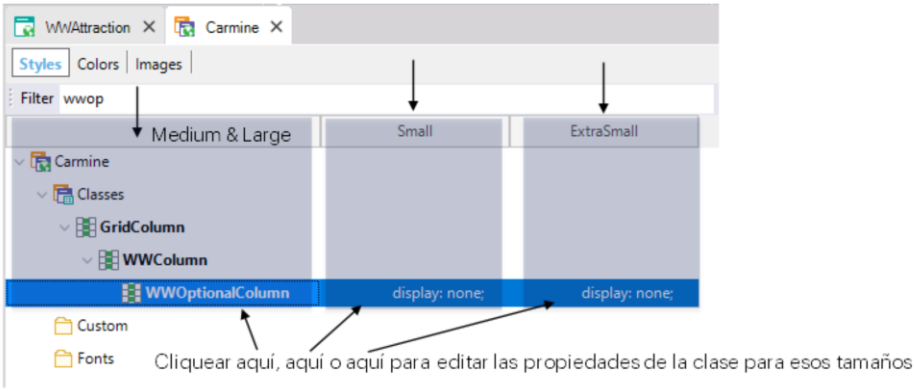
Por defecto solo se establecen dos: Small y ExtraSmall. Pero en verdad tenemos tres, puesto que los valores que vemos en la columna default corresponden a Medium y Large.

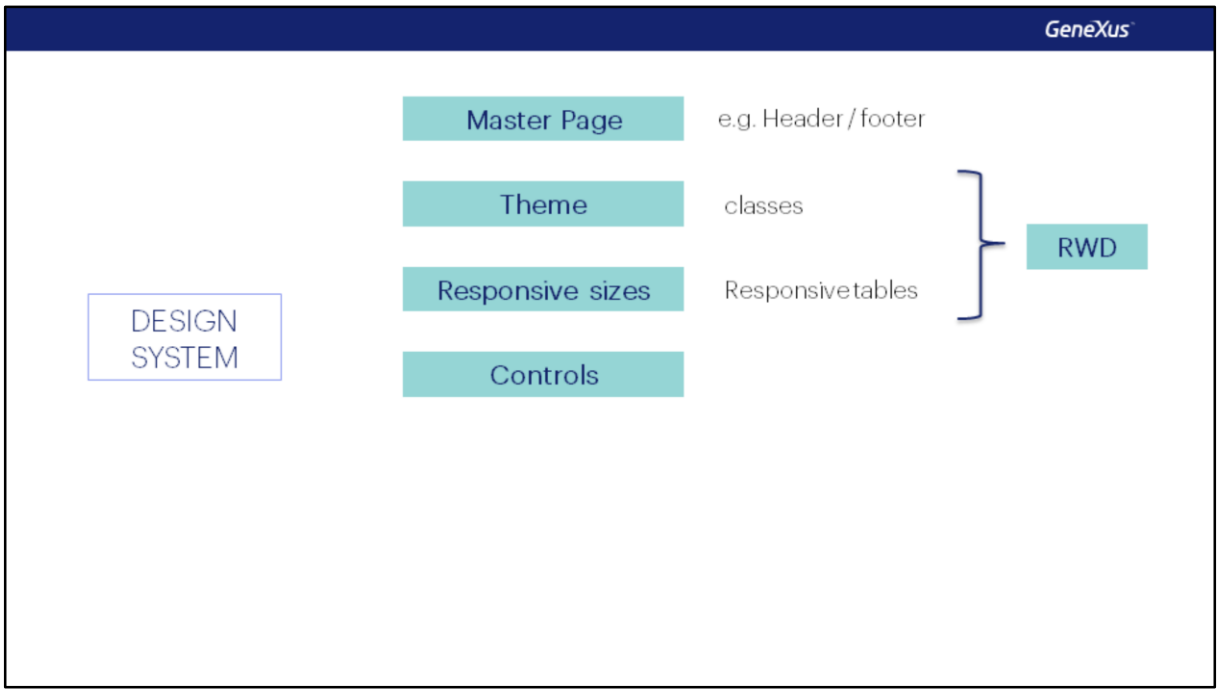
Entonces aquí podemos ver que, mientras que la propiedad Display para esta clase no dice nada, lo que significa que se mostrará en ejecución para tamaños de pantalla Medium y Large, para tamaño Small y ExtraSmall tiene el valor “none”. Y es por eso que no se muestran.

Responsive Web Design

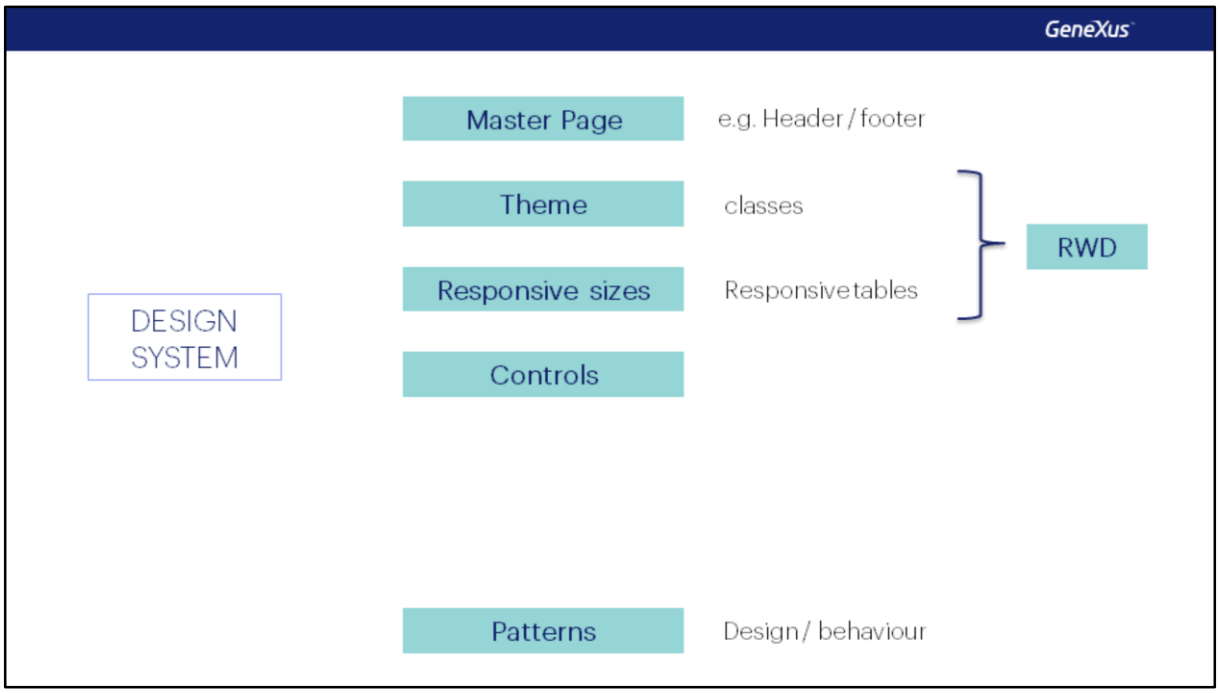
Backend

- Rules del Theme: Small, Extra-small y Default

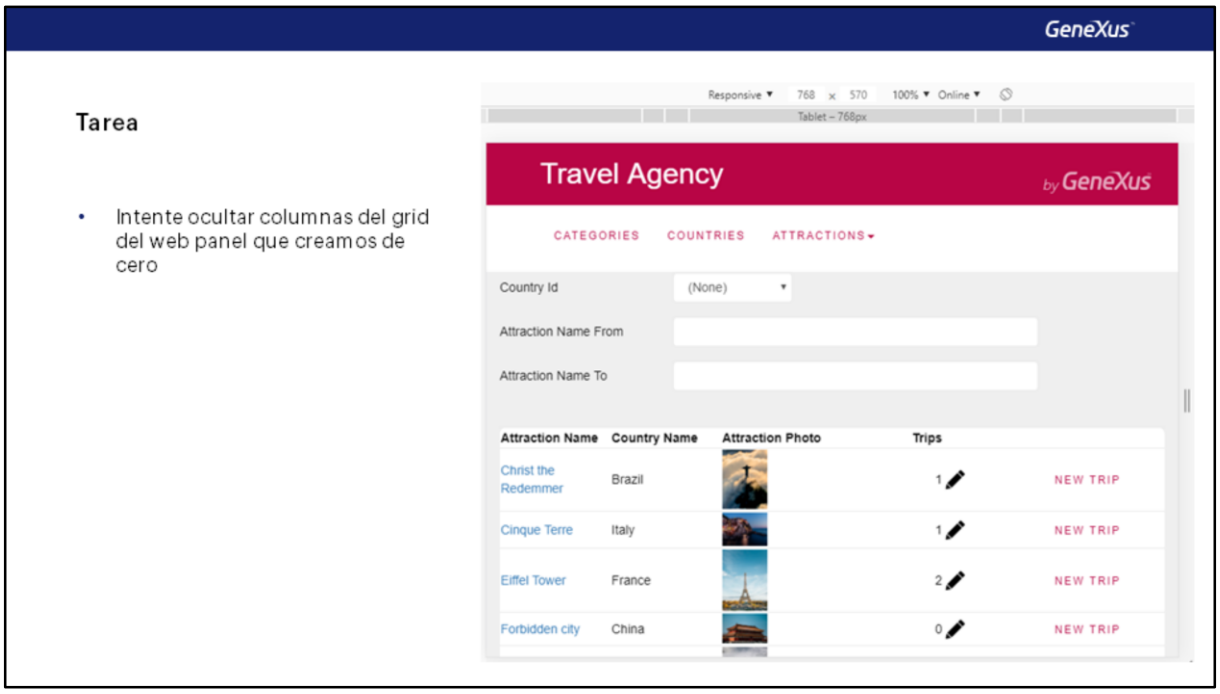




Resumiendo: otra parte importante del Design System tiene que ver con la responsividad, que se consigue tanto variando las posiciones y la visibilidad de los controles de las responsive tables, como variando las propiedades de las clases de acuerdo también al tamaño (a través de esas columnas del Theme que veíamos).

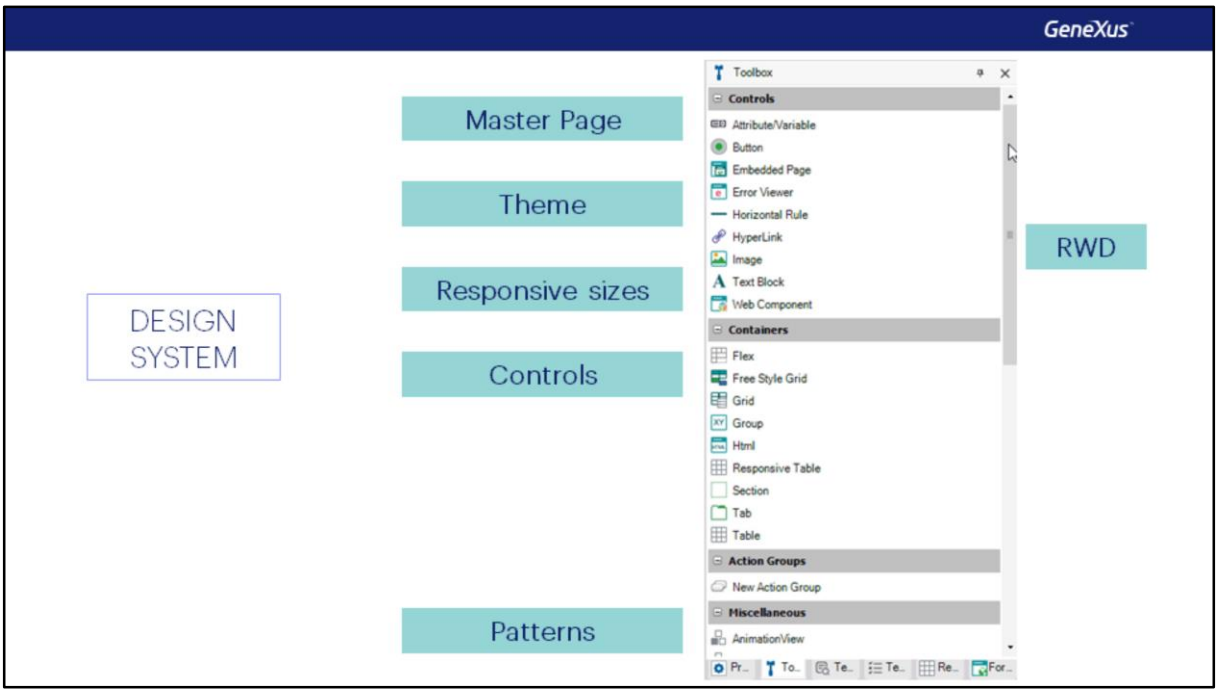


Para conseguir todo esto no tuvimos que hacer nada. El pattern Work With lo hizo todo automáticamente, en conjunción con el theme. Por eso decimos que GeneXus ya nos brinda un Design System básico, predefinido, que podemos utilizar para nuestros paneles.



Por ejemplo, ¿cómo logramos que el panel que habíamos implementado de cero tenga un comportamiento responsivo similar al del pattern? Lo invitamos a lograrlo.

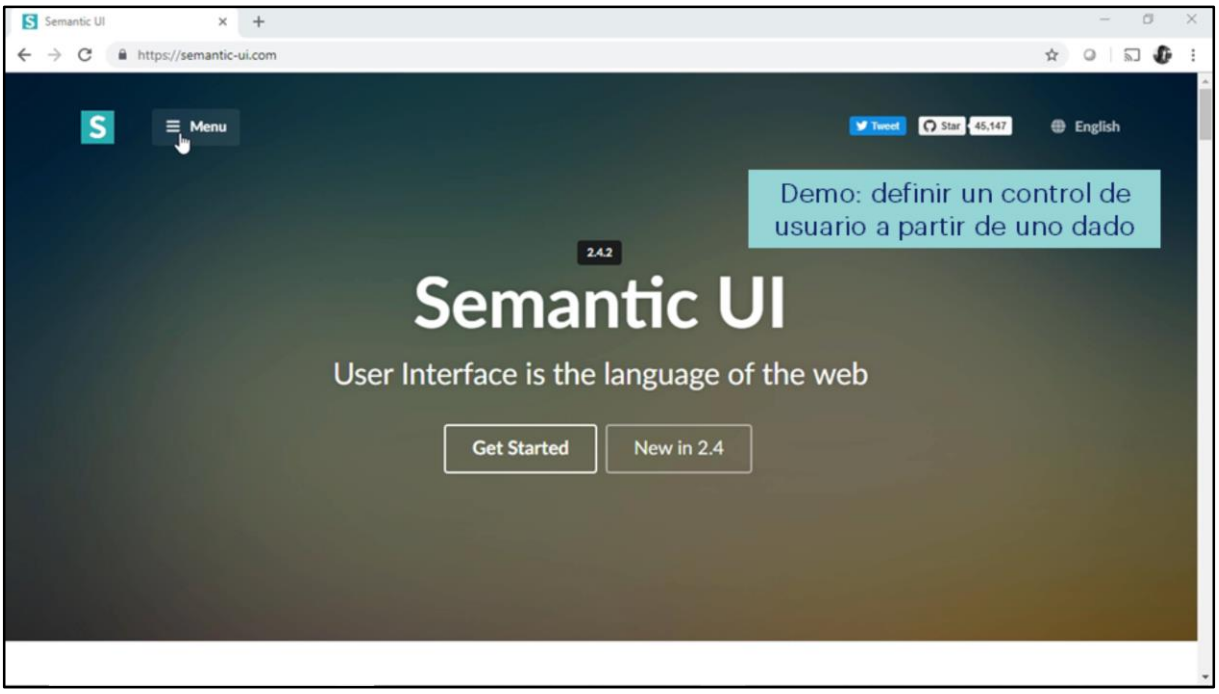
Hay mucho para decir sobre este tema. Aquí nos conformaremos con esta introducción.



Hay algunos jugadores más que, junto con todos los vistos, permiten utilizar en GeneXus un Design System potente. No los estudiaremos en este nivel del curso, pero los dejamos presentados.

Podemos no solo utilizar en nuestros forms los controles que vienen en la toolbox de GeneXus en forma predeterminada...

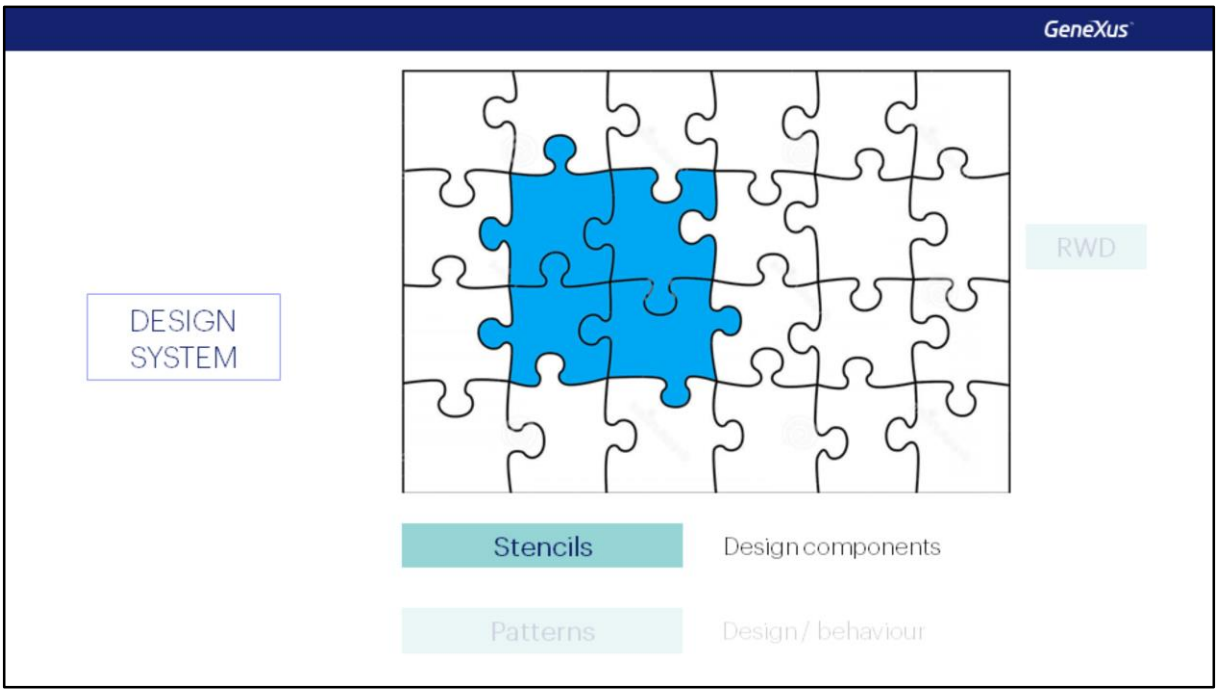
...sino que además podemos definir controles de usuario, que pueden ser copiados de plataformas que ofrecen esos controles junto con bibliotecas CSS (que son, como nuestros themes, las que especifican su estilo –el estilo de esos controles-).



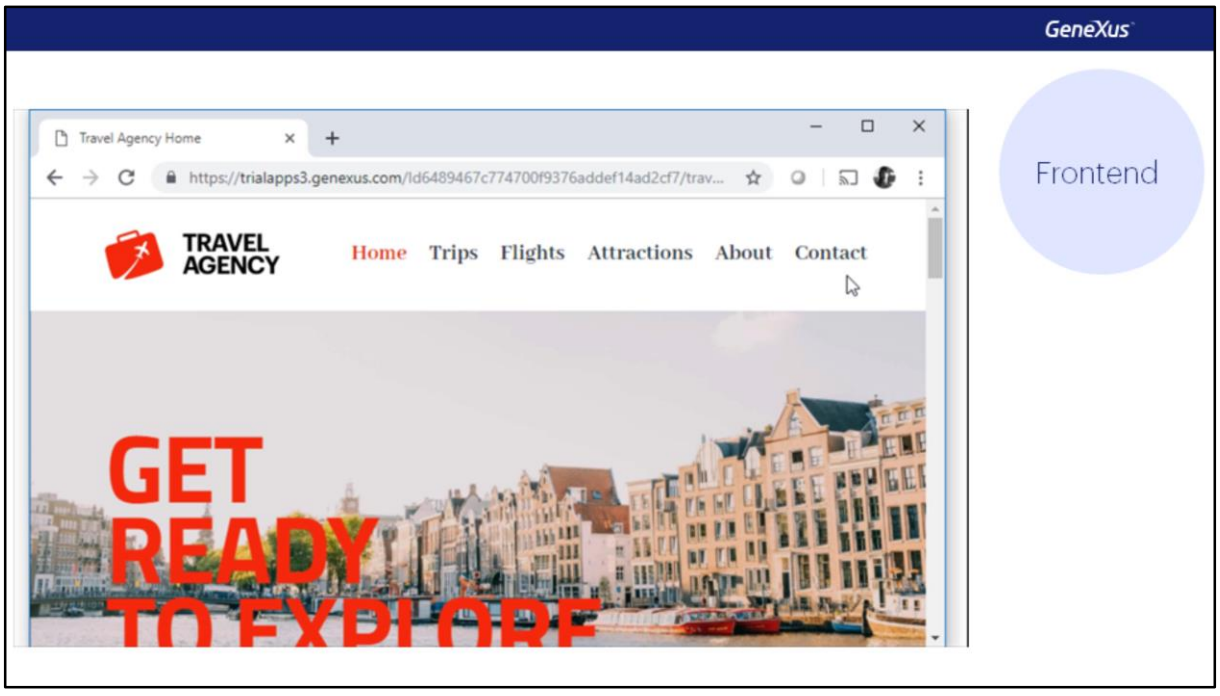
[DEMO: https://youtu.be/lKGkv7VM_ms]

Por ejemplo, esta, Semantic UI. Podríamos querer utilizar un control Card como este.

Para ello alcanzará con crear un objeto User control (le podemos llamar igual), copiar y pegar su html, cambiar los datos fijos por algo así como nombres de elementos de un SDT (para poder hacerlo dinámico, es decir, poder cargar dinámicamente ese control, con datos que vamos a poder especificar por ejemplo de la base de datos), especificar de dónde tomará el CSS, es decir, el diseño de las clases, hacer alguna cosita más y ya lo tendremos disponible en la Toolbox para utilizarlo.



Por otro lado, GeneXus ofrece Stencils, que sirven para repetir el diseño de una misma porción de pantalla (un conjunto de controles), en muchas pantallas. Es una forma de abstraer el diseño en un nivel mayor. Si podemos pensar las pantallas de una aplicación web como puzzles compuestos de piezas atómicas (que serían los controles), los stencils son una forma de agrupar un conjunto de controles cuyo diseño se va a repetir en muchas pantallas. Sería una pieza hecha de piecitas menores.



[DEMO: https://youtu.be/X4fb58cxmrA]

Mencionaremos ejemplos en el próximo video, donde situaremos estos conceptos viendo el frontend de Travel Agency.

GeneXus™

The power of doing.

More videos	training.genexus.com
Documentation	wiki.genexus.com
Certifications	training.genexus.com/certifications