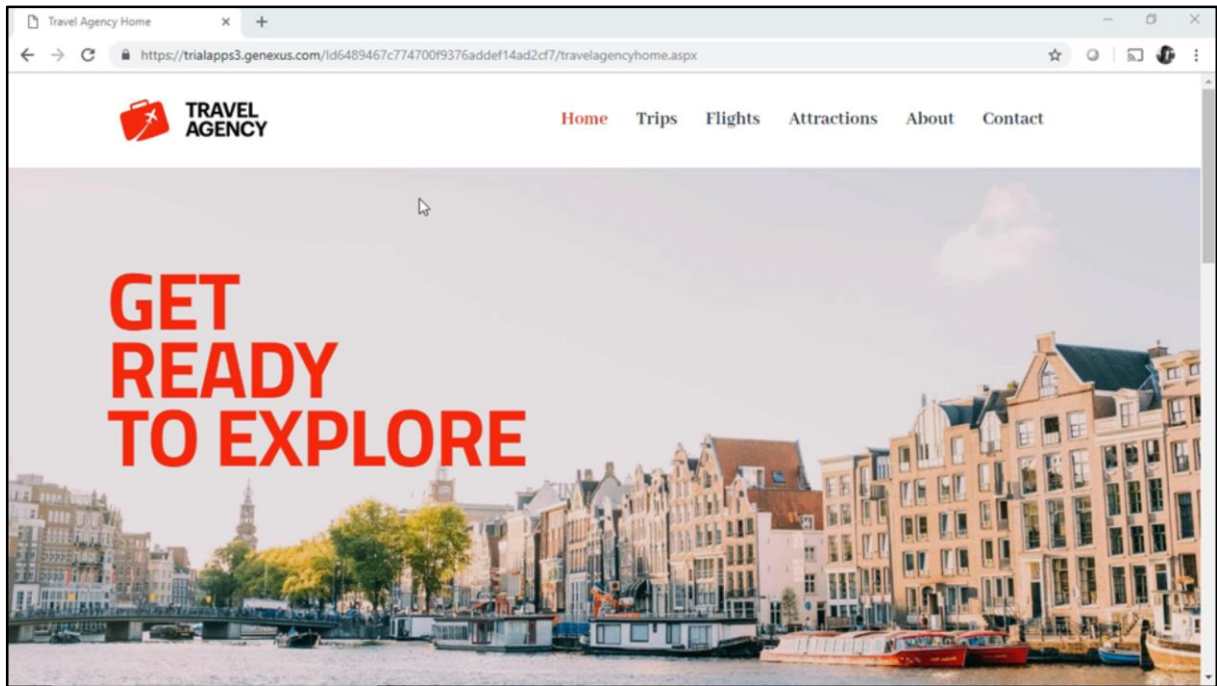


Design System en GeneXus

Frontend: SOLAPAMIENTO DE CONTROLES A TRAVÉS DE CLASES

GeneXus 16

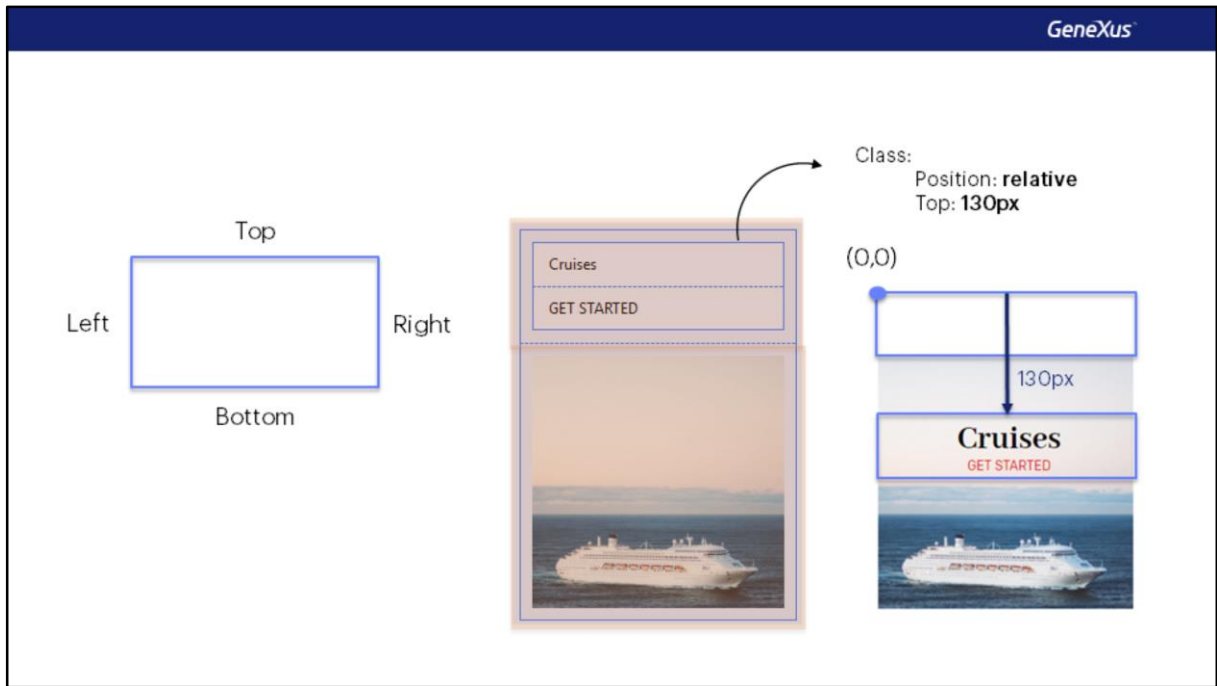


[DEMO: <https://youtu.be/hBcwzYVjtpk>]

Aquí estamos ejecutando el web panel home. Lo que hay por debajo es un archivo html con estilos asociados, que serían más o menos nuestras clases, que el navegador interpreta.

Aquí, con F12, podemos inspeccionar el html elemento a elemento y ver claramente qué clases y propiedades tiene aplicados, e incluso variarlas, para probar. Podemos, por ejemplo, ubicar un elemento en concreto de la pantalla, haciendo botón derecho/Inspect. O seleccionarlo de este otro modo.

Lo interesante es que posicionándonos sobre el elemento que deseemos, nos muestra su tamaño, espacios internos entre el elemento y el borde (padding), el borde (en este caso no tiene), el margen (tampoco). Todo eso que viene dado por los estilos (clases) que el elemento tenga asignados.



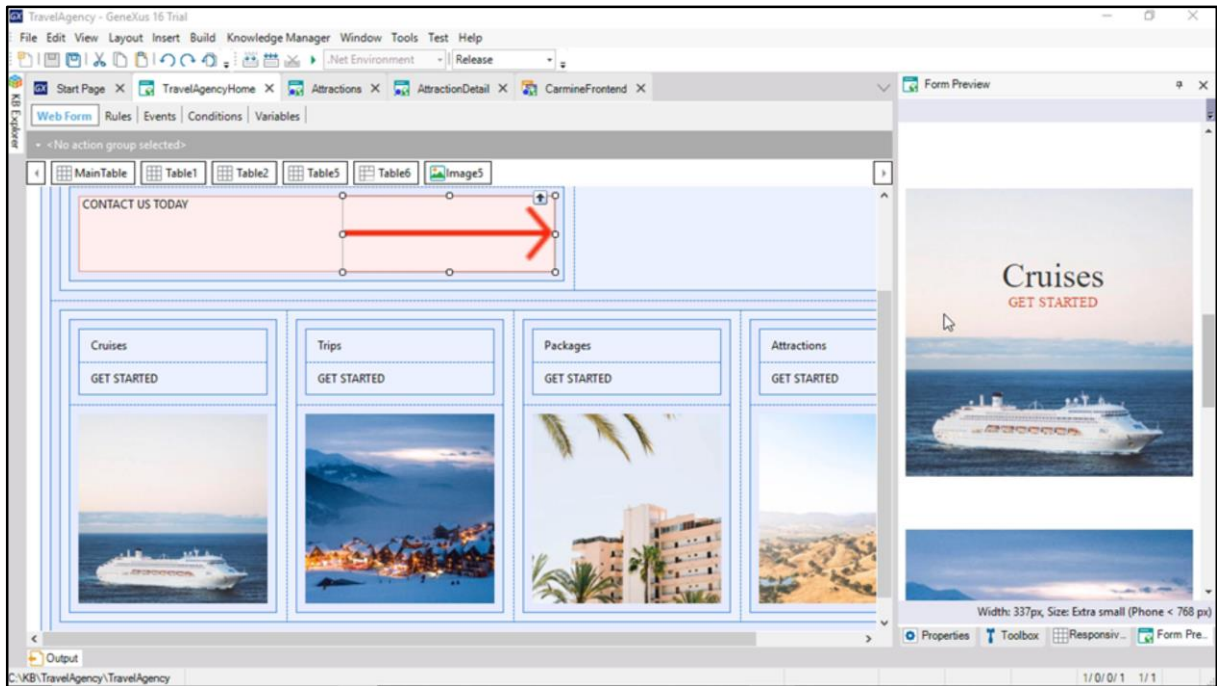
Pero como ya dijimos antes, además de establecer márgenes, bordes, paddings, alineación para el control, entre otras, las clases tienen propiedades para el posicionamiento del control, Top, Bottom, Left y Right, que permitirán, por ejemplo, solaparlo a otro.

Como en este caso. Tenemos una tabla con dos filas. En una está la tabla con los textos y en la segunda la imagen.

Lo que necesitamos es que la tabla con los textos tenga un posicionamiento diferente al que asumiría naturalmente, para que pueda solaparse sobre la imagen.

En este caso lo que hicimos fue asociarle al control que queremos mover de lugar, una clase, a la que le indicamos un posicionamiento relativo. Relativo a la posición que ocuparía naturalmente, que sería esta, arriba de la imagen, siendo el origen de coordenadas su esquina superior izquierda. Esta sería la posición (0,0). Le diremos que se mueva 130 pixeles de la posición top, superior, donde estaría naturalmente el control.

De izquierda, left y de derecha, right, lo dejamos en el mismo lugar. Y de abajo (bottom) no especificamos, porque ya lo estamos moviendo de arriba y no queremos variar el alto del control.



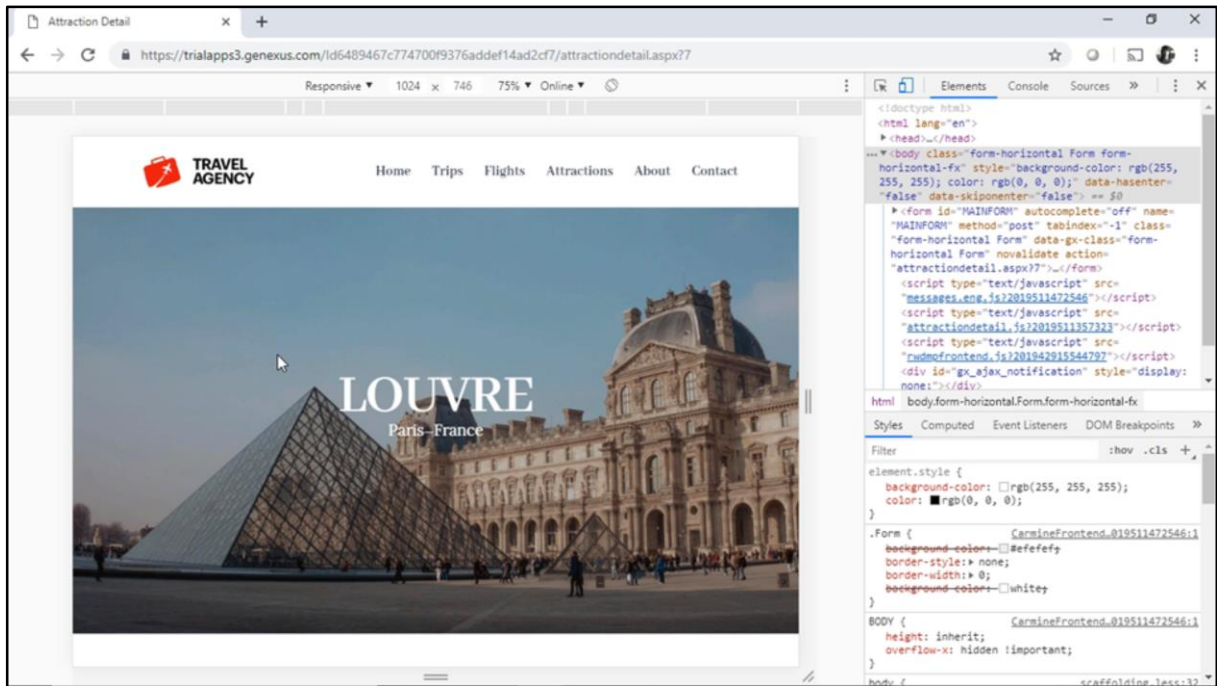
[DEMO: https://youtu.be/mZyERzrR1_0]

Lo malo de usar esta solución es que el espacio que ocuparía el control si no se moviera, se preserva, y por eso vemos este espacio en blanco aquí.

Para que ese espacio no se preserve, deberíamos utilizar posicionamiento absoluto, como hicimos en los otros dos casos de solapamiento: el del panel de atracciones, y el de detalle de una atracción –ya los veremos.

O utilizar otra opción más sencilla, que no es con posicionamiento sino con márgenes, que para no distraernos ahora, la mencionaremos al final.

Esta solución tiene el inconveniente de que al utilizar un movimiento fijo hacia abajo de 130 píxeles, no se adapta bien a la responsividad. Mientras la imagen de abajo varía su tamaño, los textos están siempre iguales, a la misma distancia de la posición de arriba.



[DEMO: <https://youtu.be/8Yn8Rb7VO-8>]

Veamos, en cambio, el caso del detalle de la atracción turística, que tendrá posicionamiento adaptable y no dejará el espacio vacío que nos molestaba.

Aquí están los textos movidos, y aquí en su lugar natural (esta vez los pusimos debajo y no encima de la imagen). Vemos la celda de alto cero, no ocupando espacio.

Aquí el web panel en GeneXus. Eliminemos provisoriamente este grid (que es el que muestra la información ampliada de la atracción), para poder visualizar bien los controles que nos interesan.

Aquí tenemos la tabla... donde colocamos el nombre de la atracción y su ciudad y país. A diferencia del caso anterior, aquí la información de los textos no es fija, se toma de la base de datos. Para la ciudad y país elegimos colocarlos dentro de un control **flex**, porque queremos que siempre queden en una fila ciudad y país pegados, sin espacios innecesarios en blanco entre ellos, independientemente del largo de su contenido. Como se toma de atributos, el contenido puede ser más o menos largo, y no queremos que deje un espacio fijo para cada uno, como pasaría al ponerlos en una tabla. Esta es la principal ventaja de un control flex. Observemos que la dirección elegida para los elementos es la default, Row, para que se muestren en una fila y no en una columna.

Bien, vamos a querer que este control con los textos no se ubique donde se ubicaría naturalmente, es decir, debajo de la foto, así que vamos a tener que modificar su posicionamiento.

Observemos que en este caso en lugar de colocar la foto y los textos en una tabla, los colocamos dentro de otro flex, esta vez con dirección Column. Podríamos haber elegido una tabla, pero nos implicaba cambiar una propiedad más (lo mencionaremos luego).

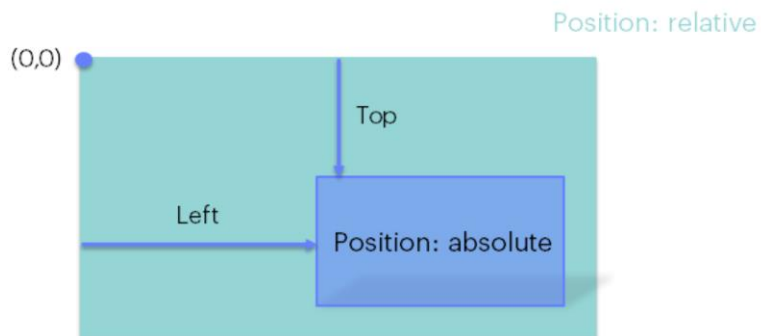
A la imagen le asignamos 600 pixels de alto, que fue lo que nos indicaron en el diseño recibido, para todos los tamaños de pantalla.

Si no modificamos el posicionamiento de la tabla con los textos, quedará debajo de la foto. Lo que hicimos fue indicarle, a través de la clase que le asignamos, `TableHeroInsideText`, que asumiera posicionamiento absoluto en relación a su contenedor, es decir, al flex donde está la foto.

Le estamos diciendo que de izquierda y derecha se pegue a los bordes del contenedor (con esto su contenido quedará centrado), y que de arriba se mueva el 40% de la altura del contenedor. Al dar el tamaño en porcentaje, será proporcional al alto del contenedor.

Un detalle engorroso pero que hay que decir: para que su punto de referencia sea el padre (este flex), debimos asignarle a él un posicionamiento relativo (aunque no lo movamos de posición respecto a sí mismo, no importa). Si no, su referente será el primer ancestro que encuentre con posicionamiento relativo. El flex, por defecto, tiene posicionamiento "static" lo que significa "el lugar donde se encuentre naturalmente".

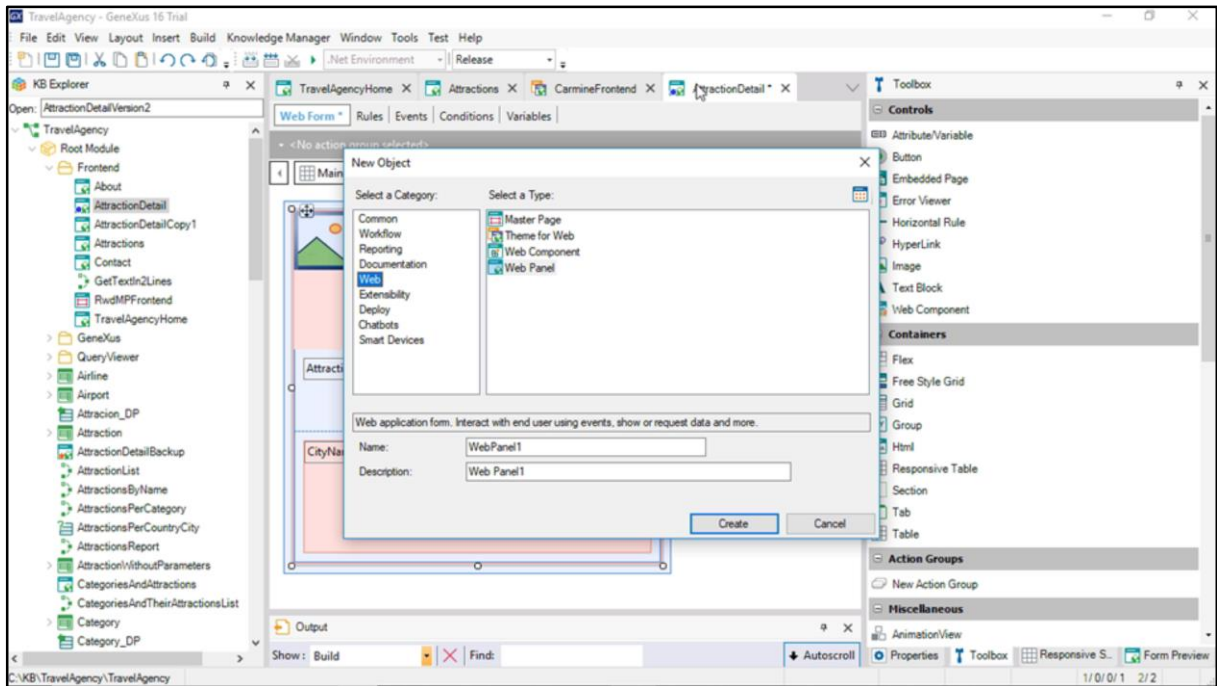
Entraremos ahora en esto para que le sirva cuando ponga manos a la obra y para explicar una mejor solución, que resuelve los defectos que señalaremos de esta que estamos viendo. Pero puede saltarlo si ahora no le interesa. A los efectos de tener una idea de la importancia de las clases y de los controles, con lo que hemos visto alcanza.



Gráficamente, la cosa sería así:

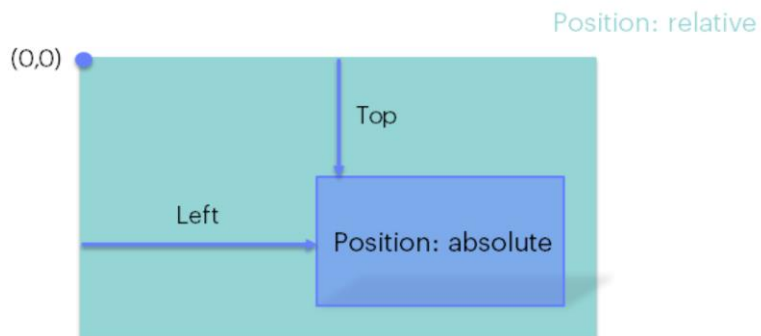
Cuando a un control se le establece posicionamiento absoluto, se quita de la capa de la pantalla donde se encontraba "naturalmente", y, se coloca en una capa superior. Como se quita, no se reserva su espacio. Al contrario de lo que ocurre cuando su posicionamiento es relativo.

¿Cual es el control en relación al cuál lo posicionamos con top, left, right y bottom?



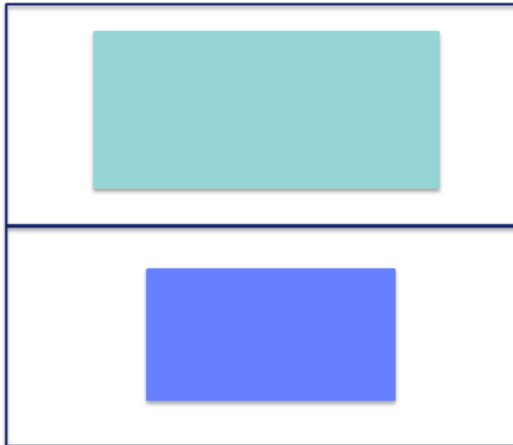
[DEMO: <https://youtu.be/AOn0Ygf0KqQ>]

Los controles están todos ordenados en una jerarquía. Cuando creamos un web panel, vemos que ya se inicializa con un control: Main table. El primer control que insertemos estará en una celda de esa tabla. Y así, iremos insertando controles en celdas hermanas de esa...
...o dentro de controles ya insertados, entonces serán hijos, nietos, bisnietos. Tendemos una verdadera familia, con abuelos, padres, hermanos.



Cuando el control tiene posición absoluta, su referente, respecto al cual se ubicará, será el primer ancestro que tenga posición relativa. Aunque este ancestro no se mueva de su lugar natural, no importa. (Si a un control le asignamos posición "relative" pero no le configuramos top, bottom, left y right, en verdad quedará donde estaba).

Responsive table



Cell:

Position: **relative** (default)

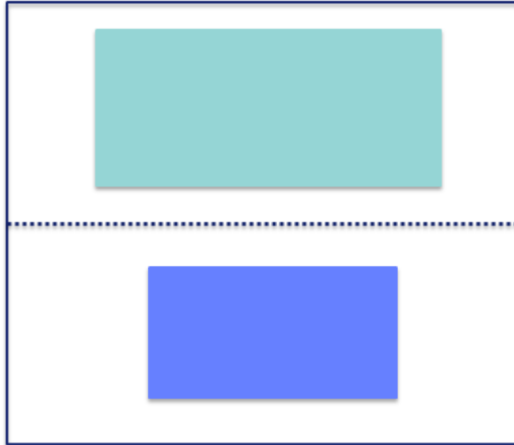
Cell:

Position: **relative** (default)

Por default, las celdas de los controles Responsive Table, tabla responsiva, tienen position relative.

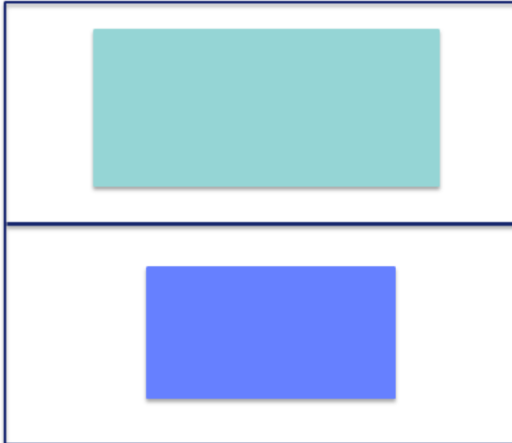
Flex

Position: static (default)

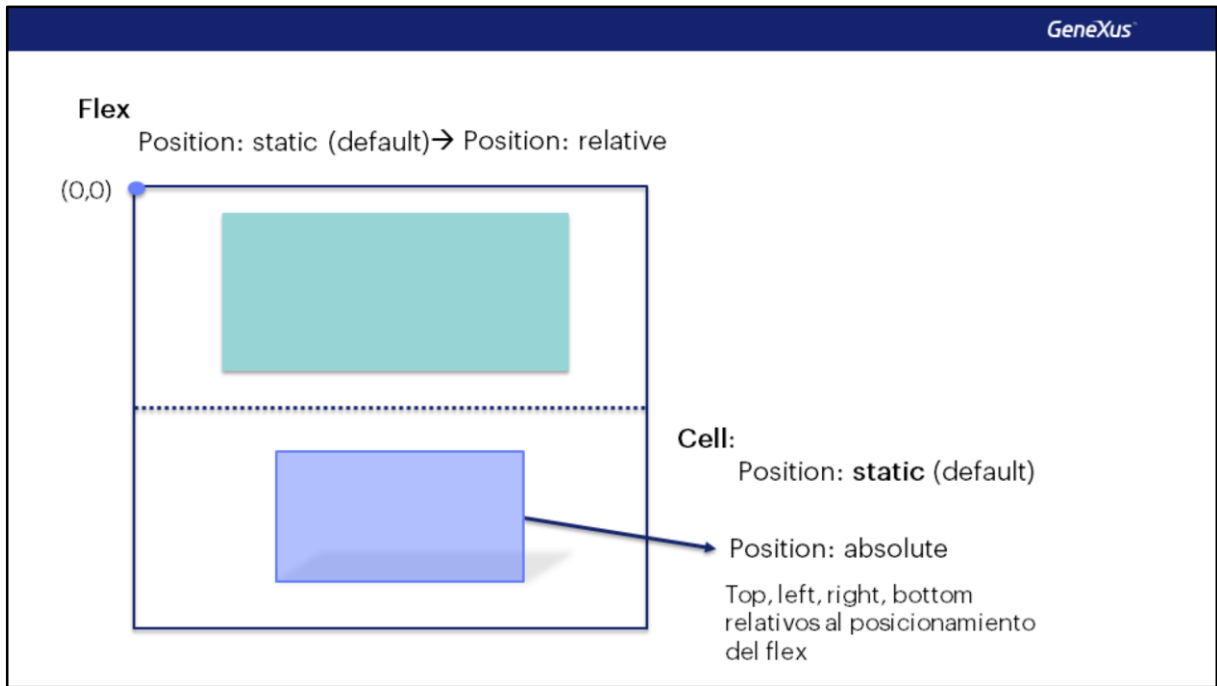
**Cell:**Position: **static** (default)**Cell:**Position: **static** (default)

Los demás: por ejemplo las celdas de los flex, tienen posición "Static".

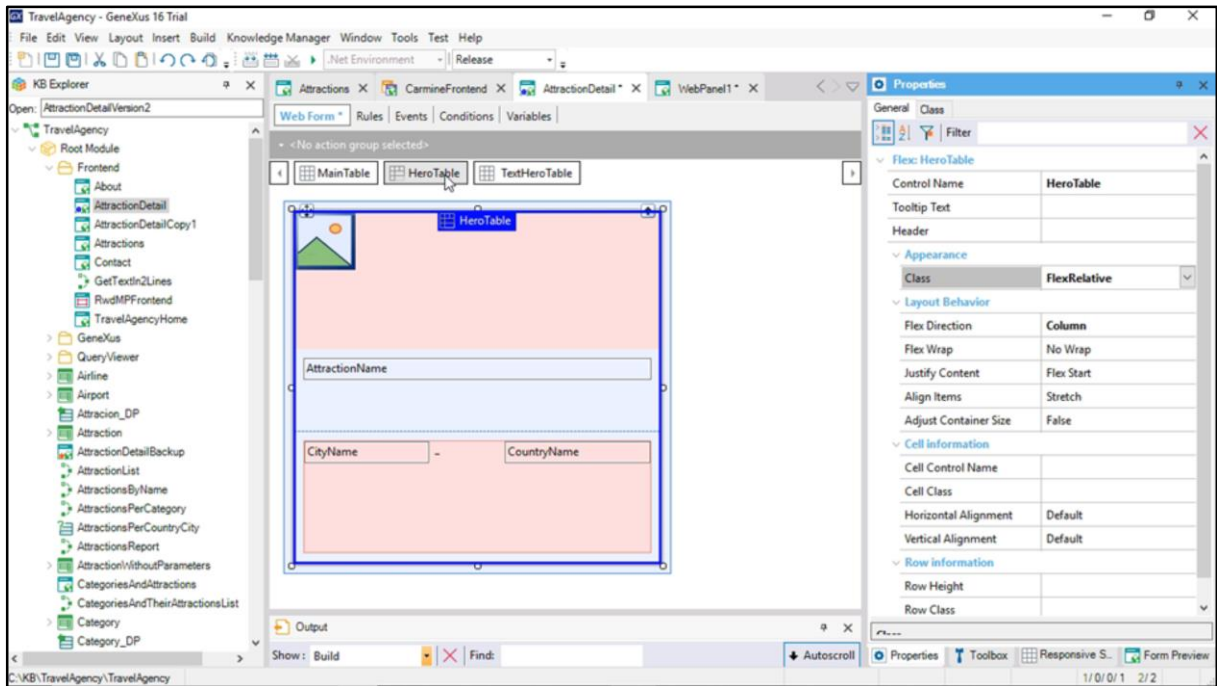
Así como el flex en sí mismo...

Responsive tablePosition: **static** (default)**Cell:**Position: **relative** (default)**Cell:**Position: **relative** (default)

... y las tablas responsivas en su totalidad. Solo sus celdas tienen posicionamiento relativo por default.

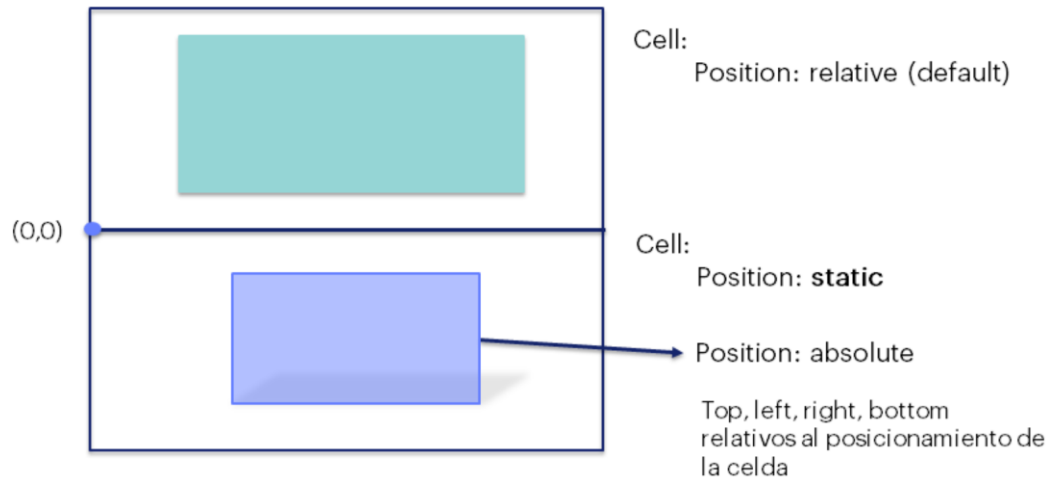


Por ese motivo, nuestra tabla con los textos (a la que le asignamos posición absoluta) no se va a ubicar respecto a su celda contenedora (que no hemos modificado en absoluto, por lo que tiene posición static), sino al flex, su padre, puesto que le hemos cambiado su posicionamiento por "relative".



[DEMO: <https://youtu.be/qMbbkTrDBhU>]

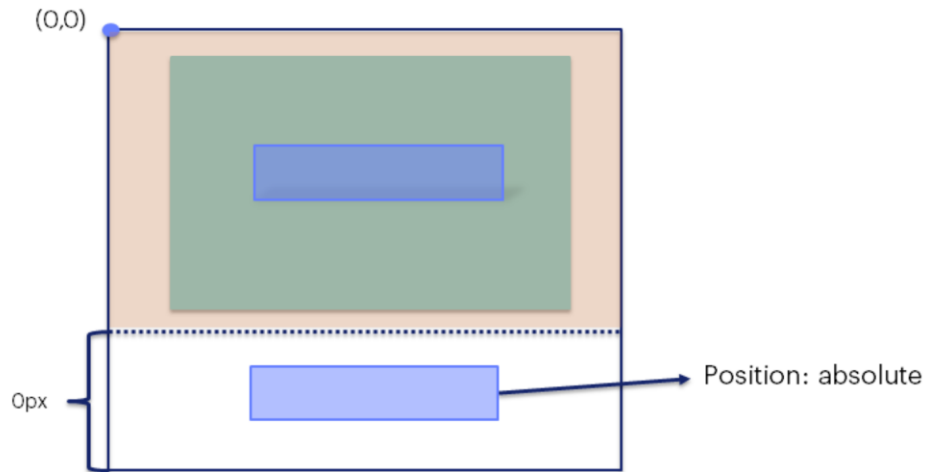
Si no hubiese sido así, igual nos habría funcionado, puesto que ese flex se encuentra dentro de una celda de una tabla responsiva, la main, que por defecto tiene posición relative. Pero es más prolijo y seguro hacerlo de este modo, que es el que de verdad queremos, ya que podríamos mover de lugar el flex para que esté dentro de otro control que tenga otro posicionamiento.

Responsive tablePosition: static (default) → Position: **relative**

Si en vez de utilizar un control flex, hubiéramos utilizado una tabla responsiva, habríamos tenido que modificar, además de la propiedad position de la tabla, la de la celda, ya que si no, el posicionamiento absoluto de nuestro control lo sería en relación a la celda y no al padre.

Flex

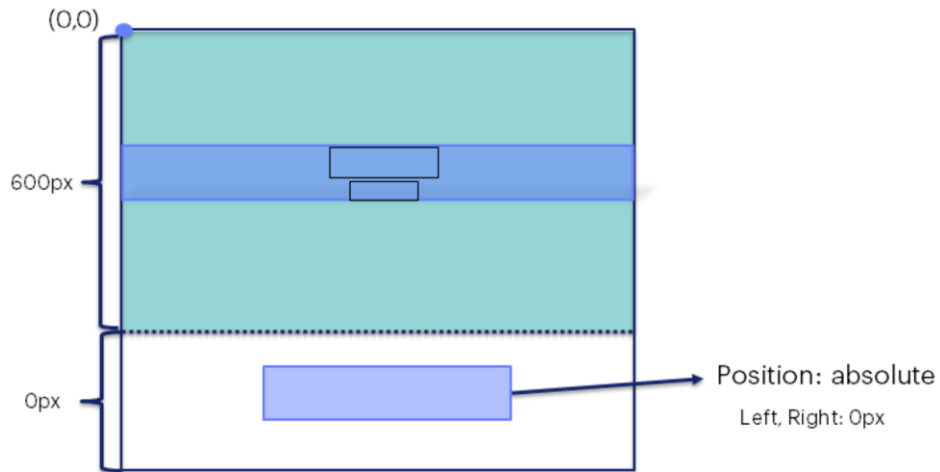
Position: static (default) → Position: relative



Volviendo a nuestro ejemplo: en verdad nosotros lo que queremos es solapar nuestro control de la celda 2 a la imagen, que está en la celda 1, y no a la tabla en su totalidad. Pero observemos que teniendo una tabla con dos celdas únicamente, y siendo la segunda una celda que quedará vacía, sin ocupar espacio, puesto que elegimos para su control un posicionamiento absoluto, el tamaño de la tabla, será igual al tamaño de la celda 1.

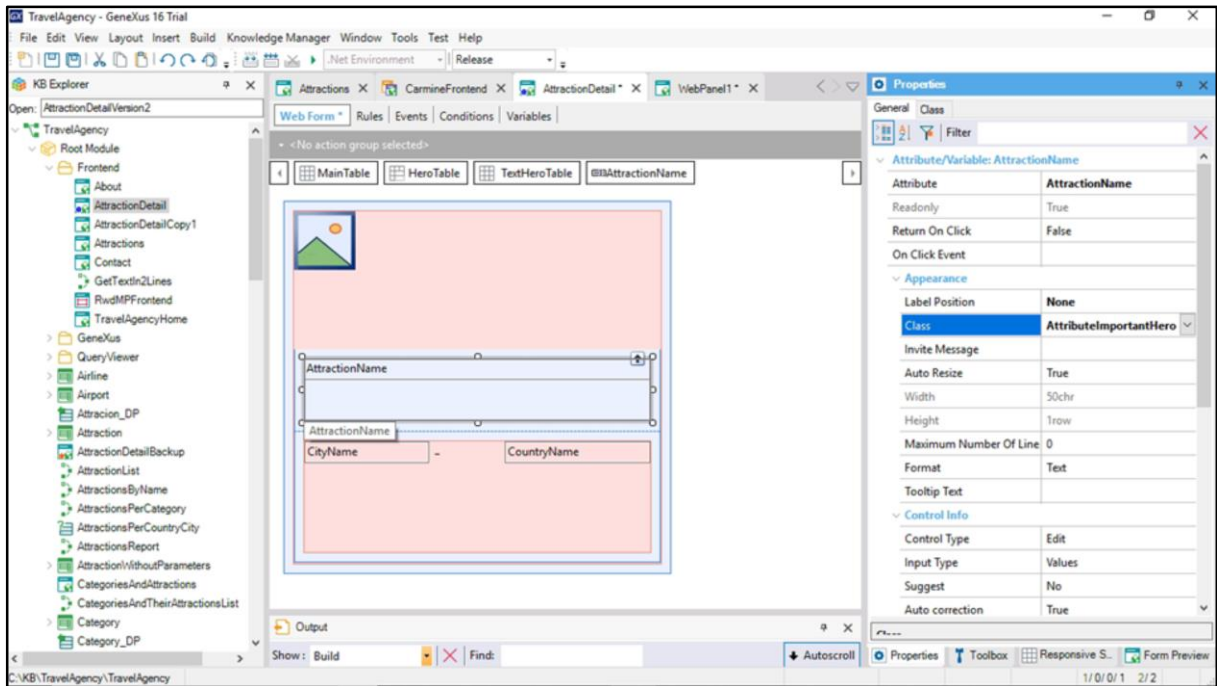
Flex

Position: static (default) → Position: relative



La celda 1 y la imagen coinciden exactamente en espacio: a la imagen le habíamos asignado un alto de 600px, y el ancho es el de la pantalla.

Por ello, al decirle a nuestro control Left y Right 0px, le estamos diciendo que se expanda hasta ocupar todo el ancho de la tabla...

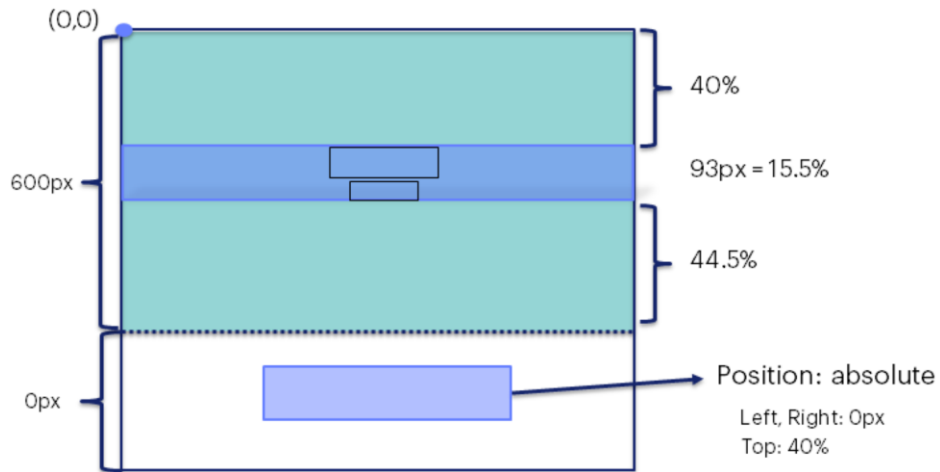


[DEMO: https://youtu.be/RleiTlSzt_k]

(observemos que además centramos horizontalmente los controles internos)...

Flex

Position: static (default) → Position: relative

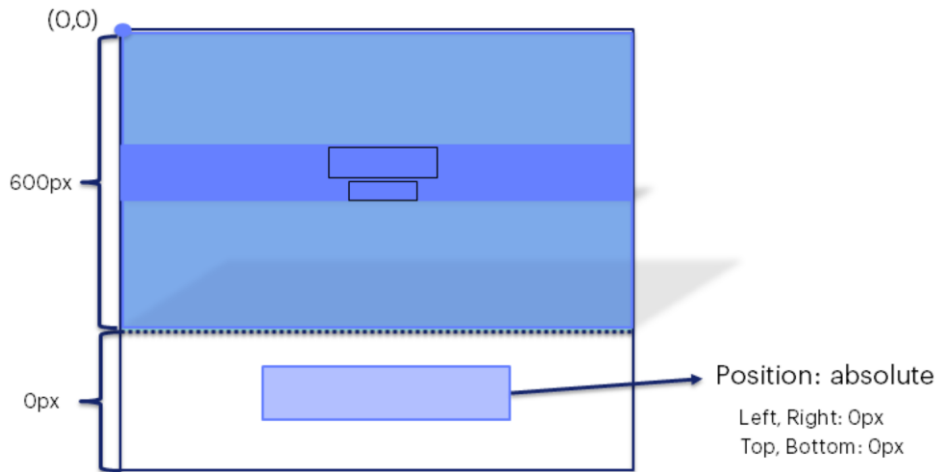


...y al decirle top: 40%, le estamos diciendo que del borde superior de la tabla, se desplace el 40% del alto total, que sabemos es de 600px independientemente del tamaño de pantalla.

¿Por qué utilizamos 40%? Porque sabemos que el ancho de los textos es más o menos de 93 pixels, que es un 15.5% de 600. Todo esto es demasiado dependiente de esos tamaños. Si los cambiáramos, se nos van a desajustar las proporciones y ya esta solución no nos va a servir.

Flex

Position: static (default) → Position: relative



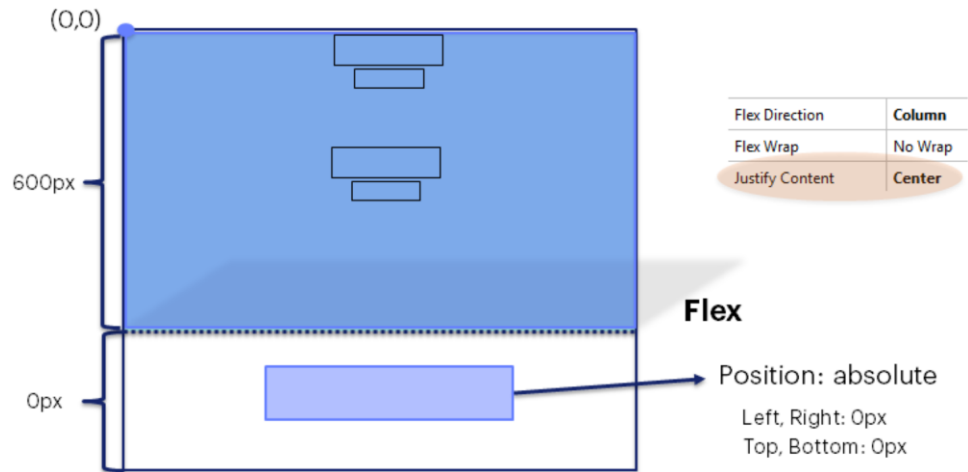
La mejor solución sería una donde el control quede siempre centrado, independientemente de todos los tamaños.

Para ello, nos alcanza con especificar también las posiciones Top y Bottom de 0px, para que el control se expanda y ocupe el mismo espacio que la imagen, que es el espacio de la celda1, que es el espacio del contenedor entero, el flex.

Para que nos queden centrados los textos también verticalmente como vemos en la imagen...

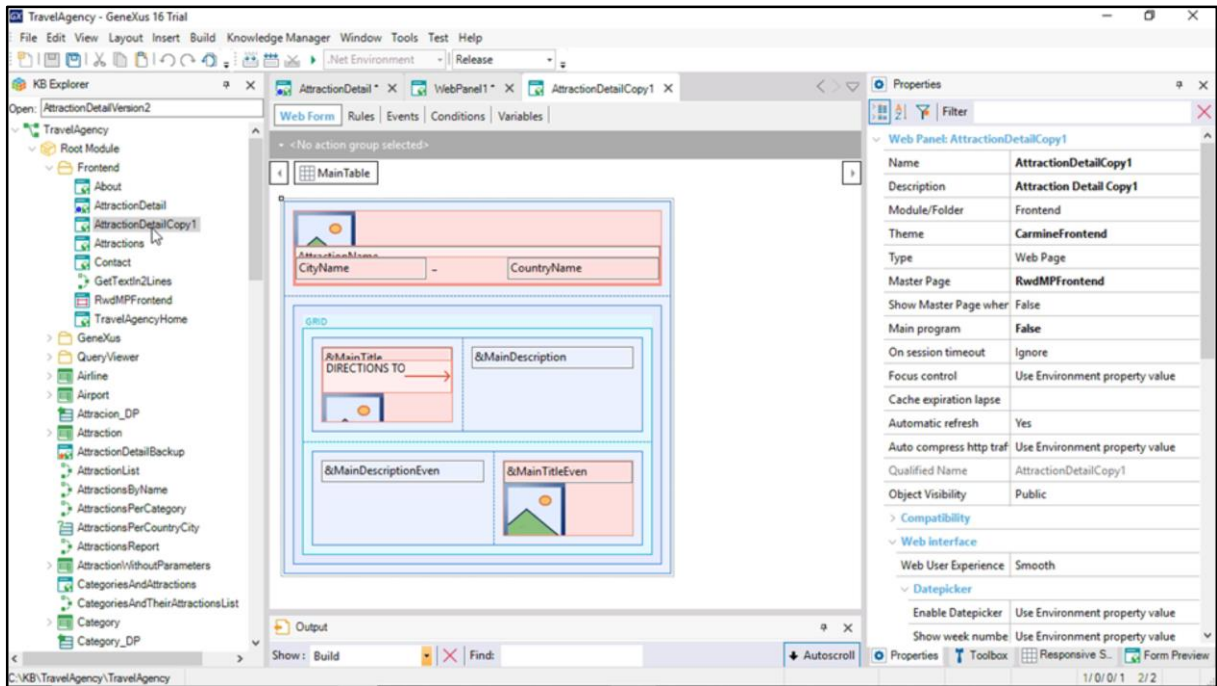
Flex

Position: static (default) → Position: relative



...y no así, también deberemos hacer algo.

Lo mejor, es utilizar un flex también para estos textos, en lugar de una tabla, con dirección Column, claro, y Justify content: center...

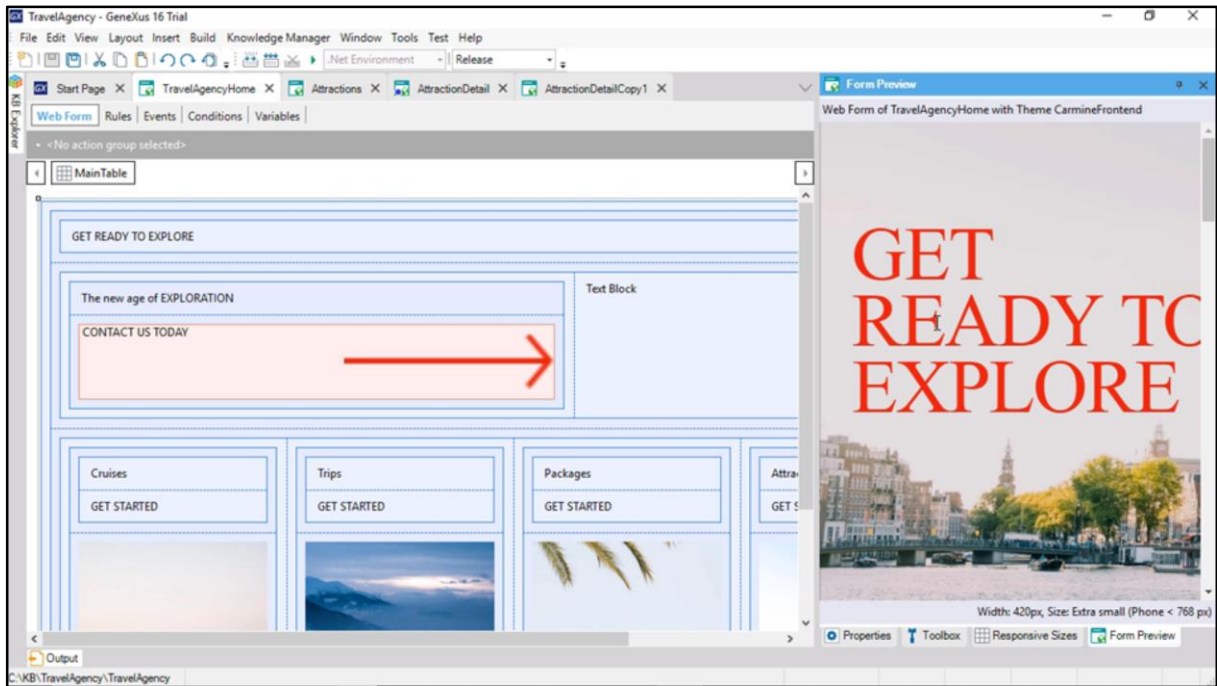


[DEMO: <https://youtu.be/urTsOwSudIQ>]

Como hicimos en esta copia del web panel...

Aquí tenemos la mejor solución de todas. Siempre, independientemente del tamaño de pantalla, del tamaño de la imagen, del tamaño de los textos, éstos quedarán centrados.

Resumiendo



[DEMO: <https://youtu.be/tn8PlmrsMNQ>]

Resumiendo lo visto hasta aquí:

Para implementar el solapamiento de uno o varios textos sobre una imagen...

Si la imagen es fija, una opción fue la que utilizamos aquí, de definirla como background image en la clase de la tabla contenedora del text block.

La segunda opción que vimos, sirve tanto para datos fijos como tomados de la base de datos: fue la de cambiar el posicionamiento de la tabla con los textos de manera relativa. Relativa a la posición que ocuparía naturalmente. Vimos dos desventajas, que se resolvían con posicionamiento absoluto. Una, la que ahora nos importa, era que se preservaba el espacio natural del control. Allí hablamos de otra opción más sencilla para esto mismo, que mencionaríamos al final y que era con márgenes y no con posicionamiento.

Es la que implementamos en esta copia del panel home, en la que invertimos el orden en que aparecen imagen y textos. A los textos le asociamos otra clase, en la que en vez de cambiar el posicionamiento mediante las propiedades correspondientes, le colocamos un margen superior (Margin top) negativo. El margen de un control es el espacio que lo rodea. Si le asignamos un valor negativo, el control se moverá en el sentido contrario en el que lo haría naturalmente (con un margen positivo).

Si se quiere que suba, le asignamos margen superior negativo, si queremos que baje, margen inferior negativo, y así.

Aquí tuvimos que invertir el orden de los controles, porque necesitamos que el que estamos moviendo utilizando el margen quede encima del otro. Los que están más arriba quedan siempre debajo de los que vienen luego.

No entraremos en detalle, pero aquí la propiedad z-index que se utiliza para indicar justamente el orden de las capas en caso de solapamiento (cuál quedará encima de cuál) no tiene efecto, porque se trata de dos controles hermanos. Si quiere ahondar sobre esto, vea aquí: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index.

Esta solución, sin embargo, sigue teniendo el problema de que no nos permite adaptarnos a la responsividad. Usar posicionamiento es mejor cuando tenemos que movernos un porcentaje respecto a un contenedor. Y esa fue la última solución que vimos. Allí utilizamos posicionamiento absoluto, lo que permitía que el control a solapar sobre otro se moviera de acuerdo a la posición del contenedor de ambos (que debe tener posición relative. Como nota: si a un control le asignamos posición relative, pero no modificamos ninguna de sus propiedades Top, Bottom, Left y Right, lógicamente queda donde está).

Esto intentó ser una pequeña muestra de la importancia de las clases (y también de los controles, claro) para el diseño de nuestra aplicación.



Videos

training.genexus.com

Documentation

wiki.genexus.com

Certifications

training.genexus.com/certifications