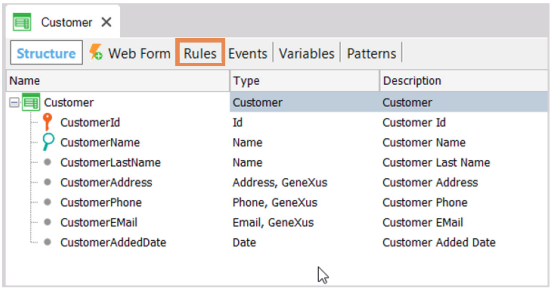
Reglas: Repaso y Client-Side Validation

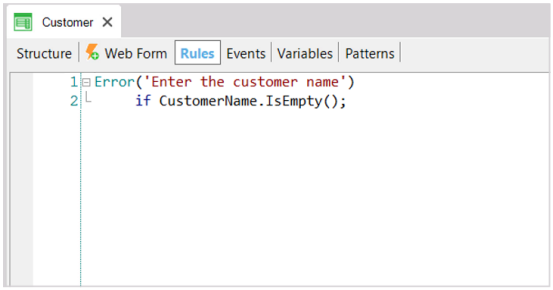
GeneXus™

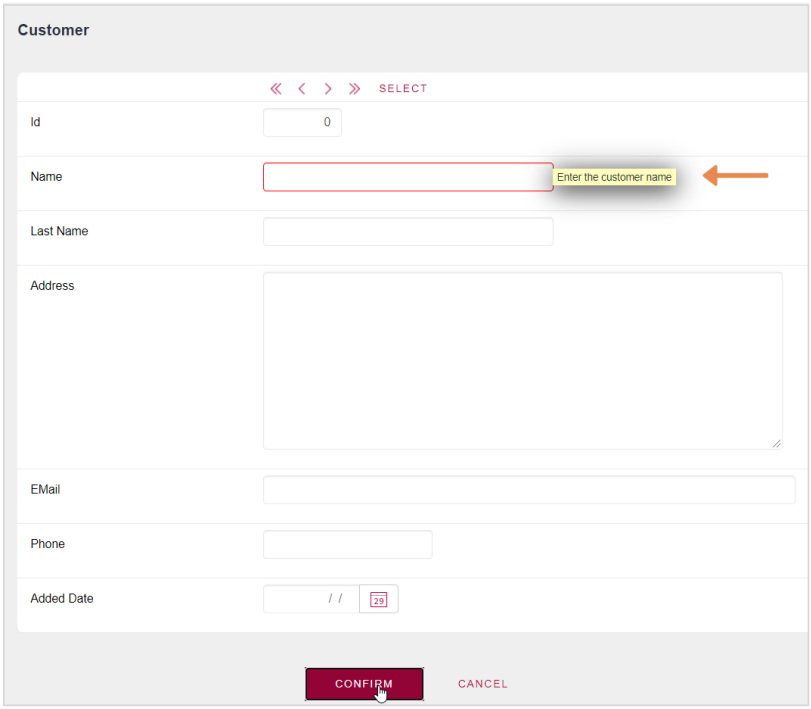
Rules



En el curso anterior aprendimos que el objeto Transacción cuenta con una sección llamada **Rules**, en la cual son definidos los controles que se deben efectuar o las reglas que se deben cumplir para una cierta realidad.

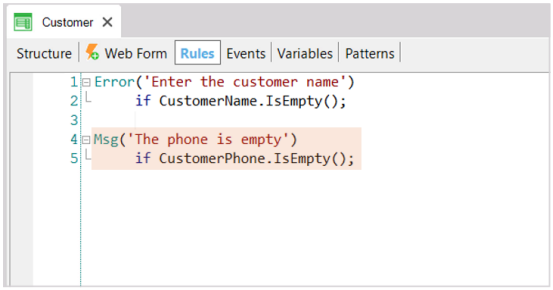
Rules

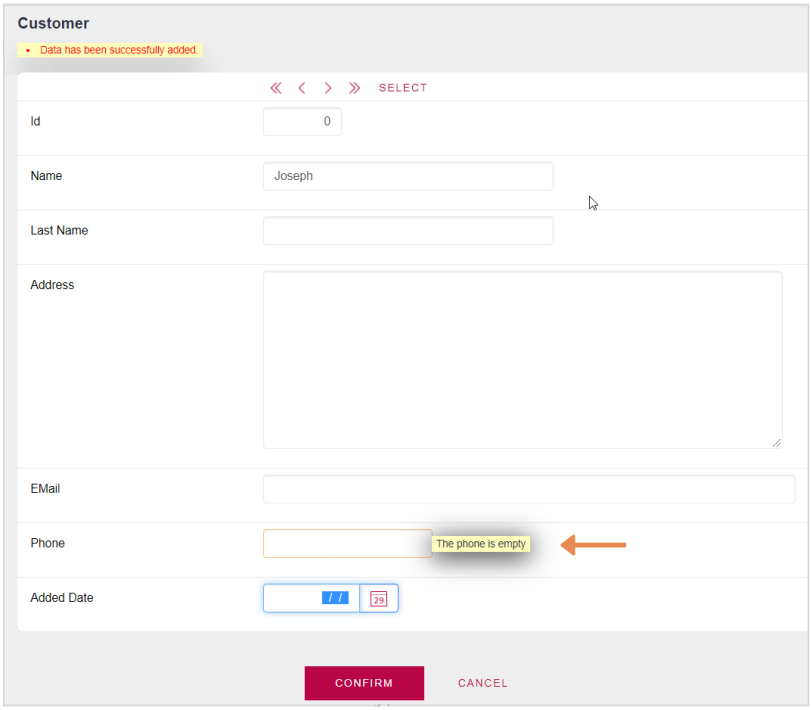




Nos centramos en la regla Error, que impide que se almacene en la base de datos un registro mientras cierta condición se cumpla: por ejemplo, si estamos ingresando un cliente y dejamos su nombre vacío, ese cliente no se podrá insertar hasta que el usuario haya escrito su nombre.

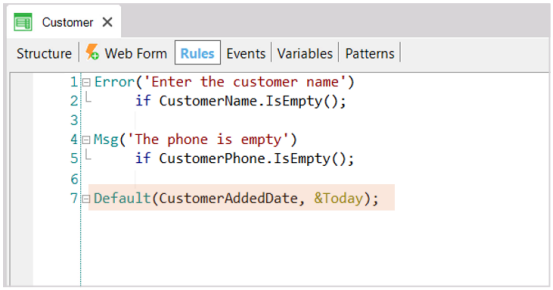
Rules

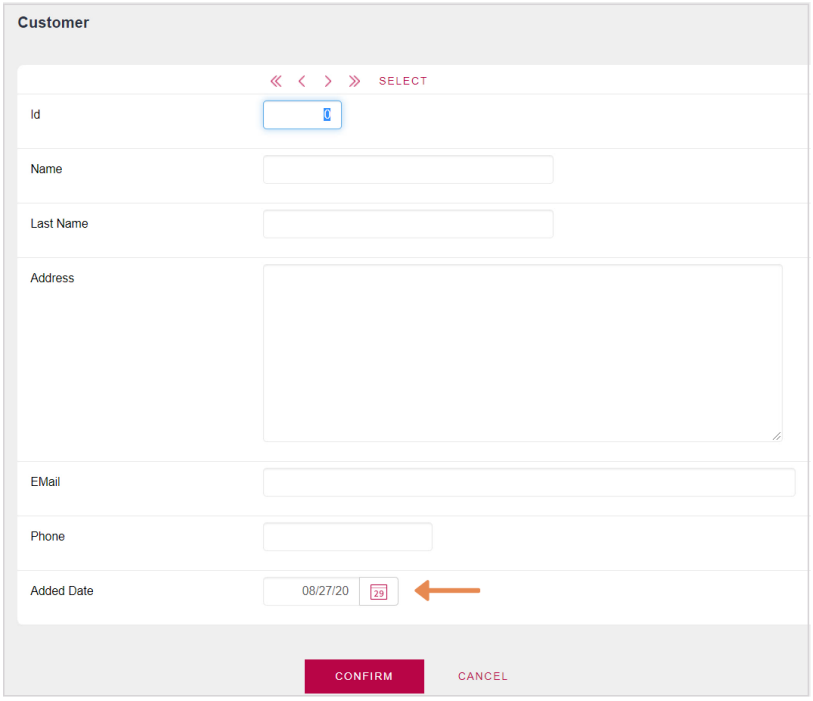




Vimos también la regla **Message**, la cual únicamente informa al usuario mediante un mensaje pero permite efectuar la grabación;

Rules





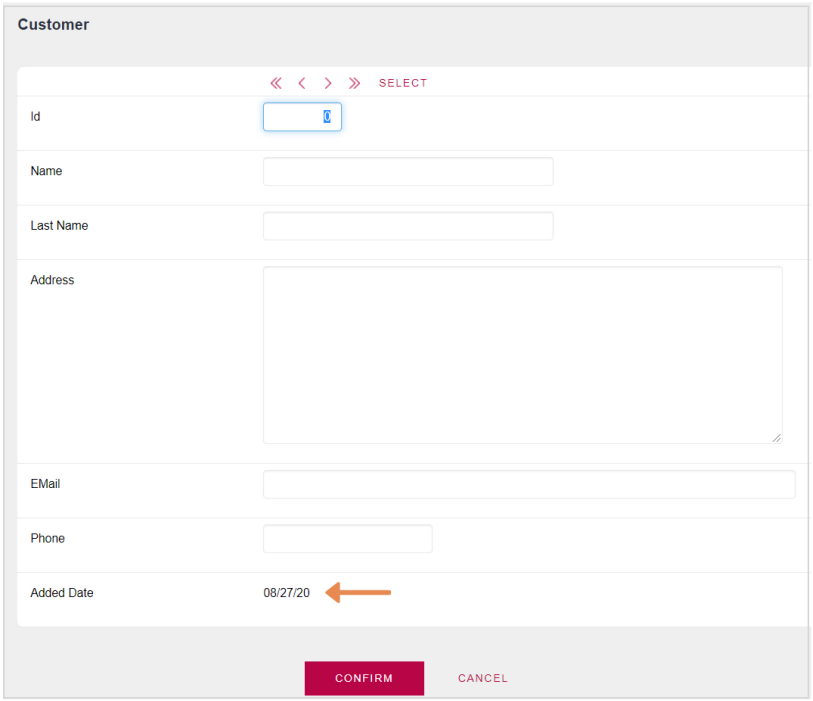
la regla **Default**, que nos permite inicializar un atributo o variable con un valor cuando accedemos a la transacción en modo de inserción;

Rules

Customer

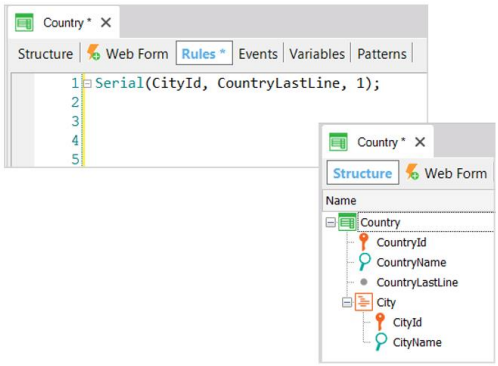
Structure | Web Form | Rules | Events | Variables | Patterns

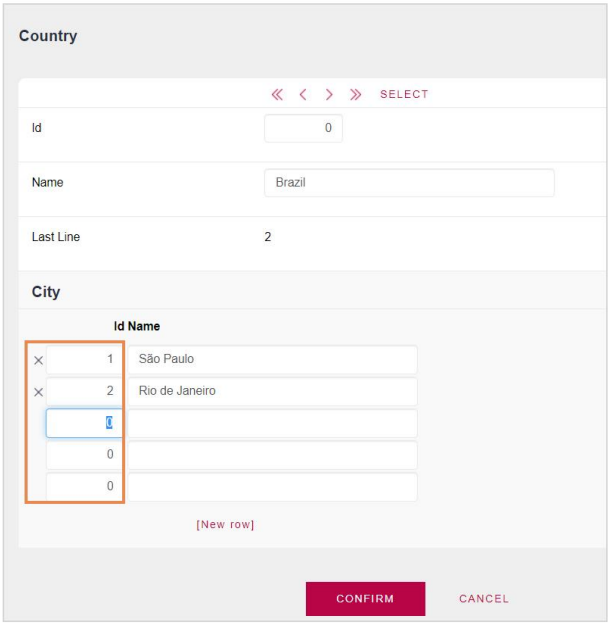
```
1  Error('Enter the customer name')
2      if CustomerName.IsEmpty();
3
4  Msg('The phone is empty')
5      if CustomerPhone.IsEmpty();
6
7  Default(CustomerAddedDate, &Today);
8
9  Noaccept(CustomerAddedDate);
```



la regla **NoAccept**, que impide que el usuario pueda modificar un campo en el formulario: lo muestra deshabilitado,

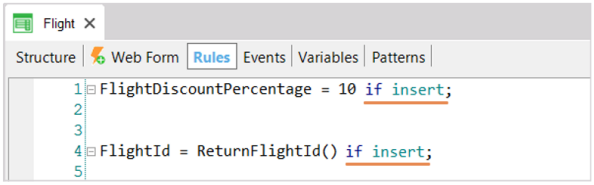
Rules





y por último la regla **Serial**, que sirve para autonumerar un segundo nivel, o tercero, u otro nivel anidado de una transacción.

Rules



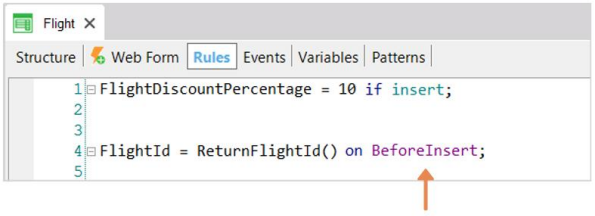
if insert

if update

if delete

Estudiamos también que mediante las reglas podemos definir asignaciones de valores o invocar objetos, y que también podemos condicionarlas para que se ejecuten solo cuando se está insertando, modificando o eliminando.

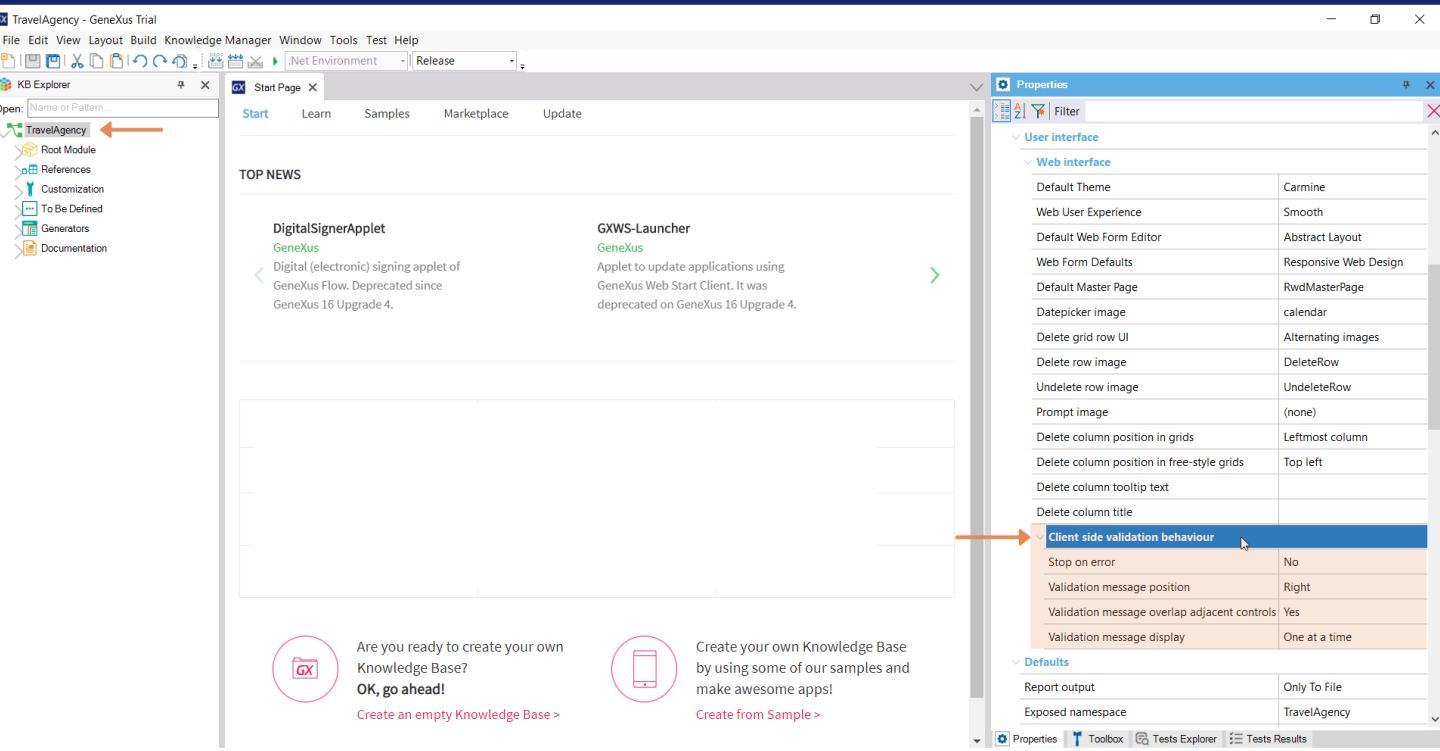
Rules



Además, si el momento elegido por GeneXus para ejecutar una regla no es el que necesitamos, podemos indicarle en qué momento preciso queremos que se ejecute, utilizando los eventos de disparo.

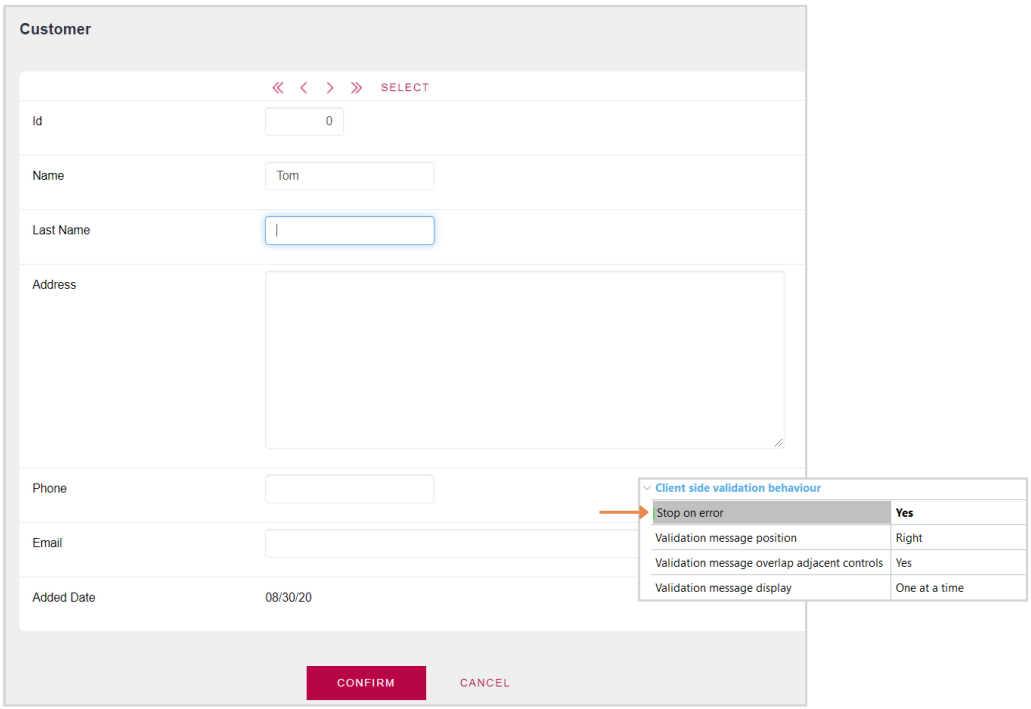
Execution of rules

Recordemos que todas estas reglas que hemos estudiado, son validadas tanto en el cliente web como en el servidor.
La validación que ocurre en el cliente es llamada Client-Side Validation, y su objetivo es brindar una buena experiencia al usuario, haciéndole sentir que la aplicación está todo el tiempo interactuando con él; pero es el servidor quien realmente valida que toda la información enviada sea consistente y no viole la seguridad del sistema, y es el único que puede operar sobre la base de datos.



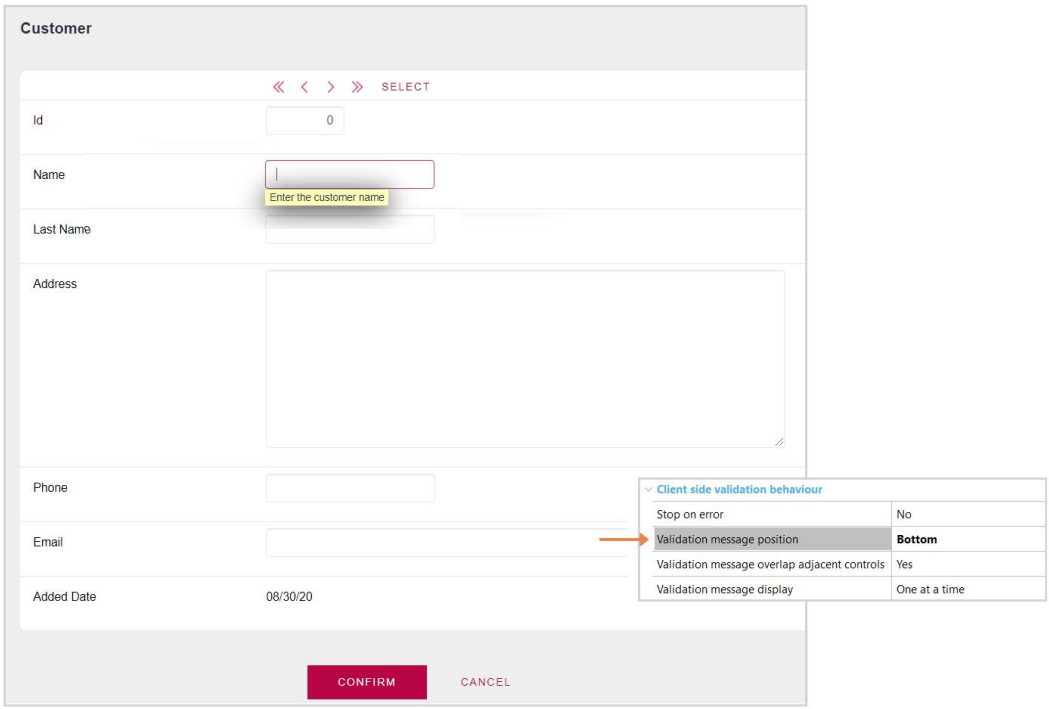
Hasta aquí hemos visto el comportamiento por defecto de estas reglas. Pero es interesante saber que contamos con un grupo de propiedades llamado **Client Side Validation Behaviour** que encontramos a nivel de la versión, el cual nos permite personalizar el comportamiento y los mensajes de las reglas de muchas formas, permitiendo que la aplicación sea más atractiva para usuarios finales.

Stop on error property



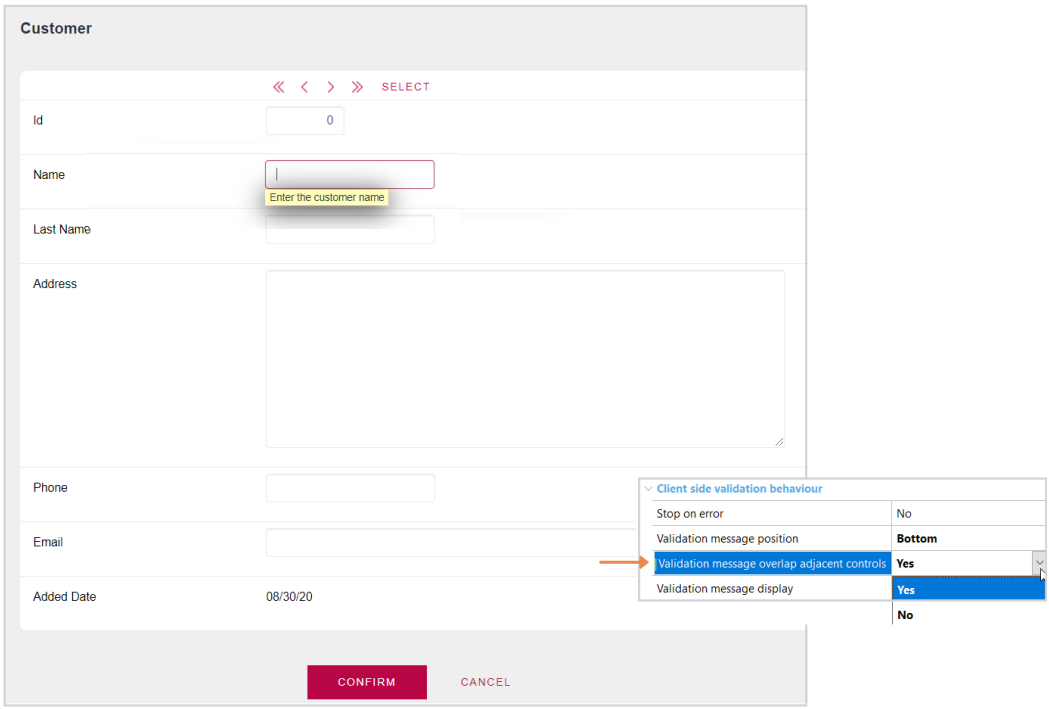
Cuando utilizamos la regla Error, vimos que aunque la transacción no permite grabar, sí permite pasar al siguiente campo, luego de haber mostrado el mensaje de error. Si nos interesara que en caso de dispararse un error, se mantenga el foco en el control y se obligue al usuario a corregir el valor para poder pasar al siguiente campo, debemos cambiar el valor de la propiedad *Stop on error* para *Yes*.

Validation message position property



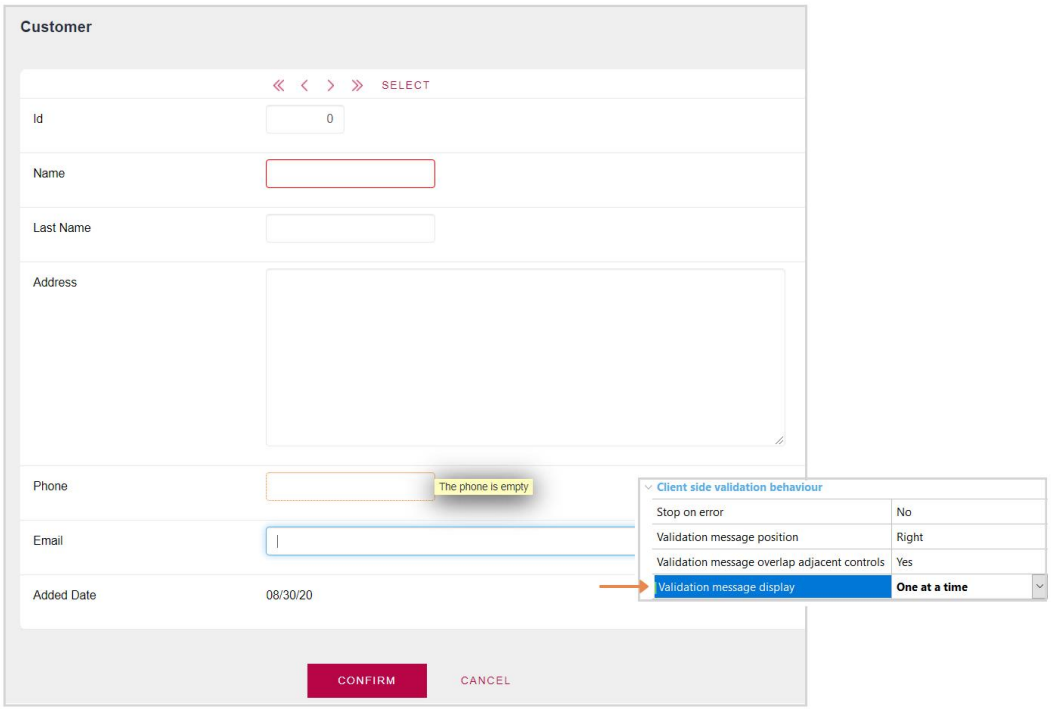
También podremos definir el lugar donde va el mensaje con respecto al campo, utilizando la propiedad ***Validation Message Position***, que cuenta con los valores *Right* (que es el valor por defecto), *Left*, *Top* y *Bottom*.

Validation message overlap adjacent property



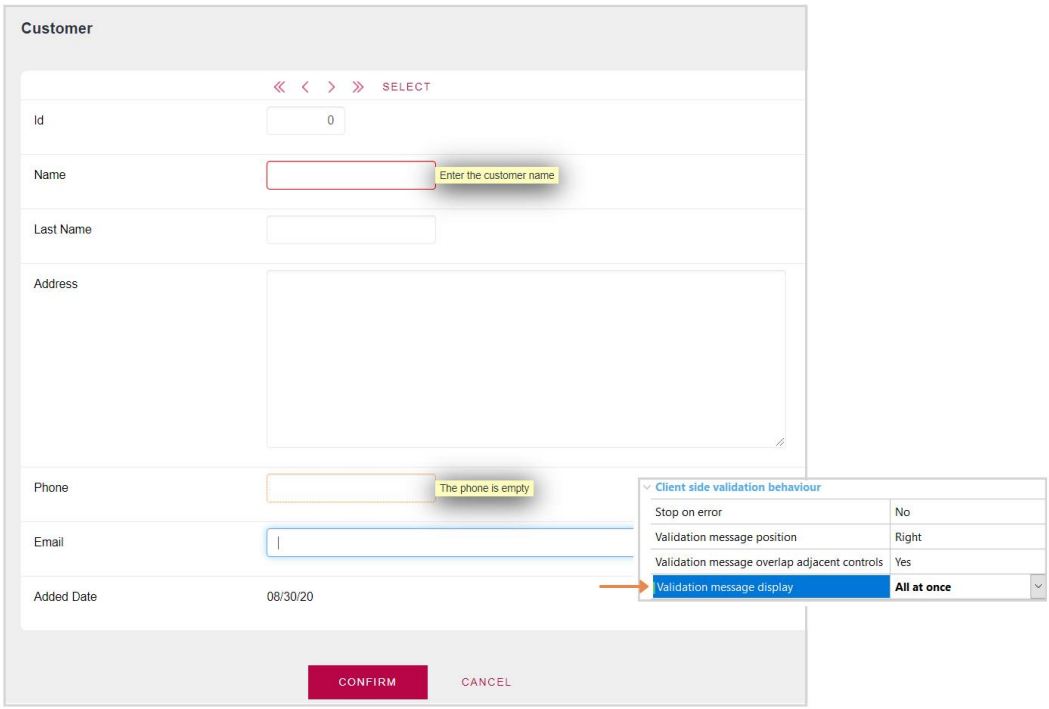
En caso de que el mensaje se muestre solapado a algún otro control, también debe configurarse la propiedad ***Validation message overlap adjacent controls***, que controla si los mensajes deben superponerse o no.

Validation message display property

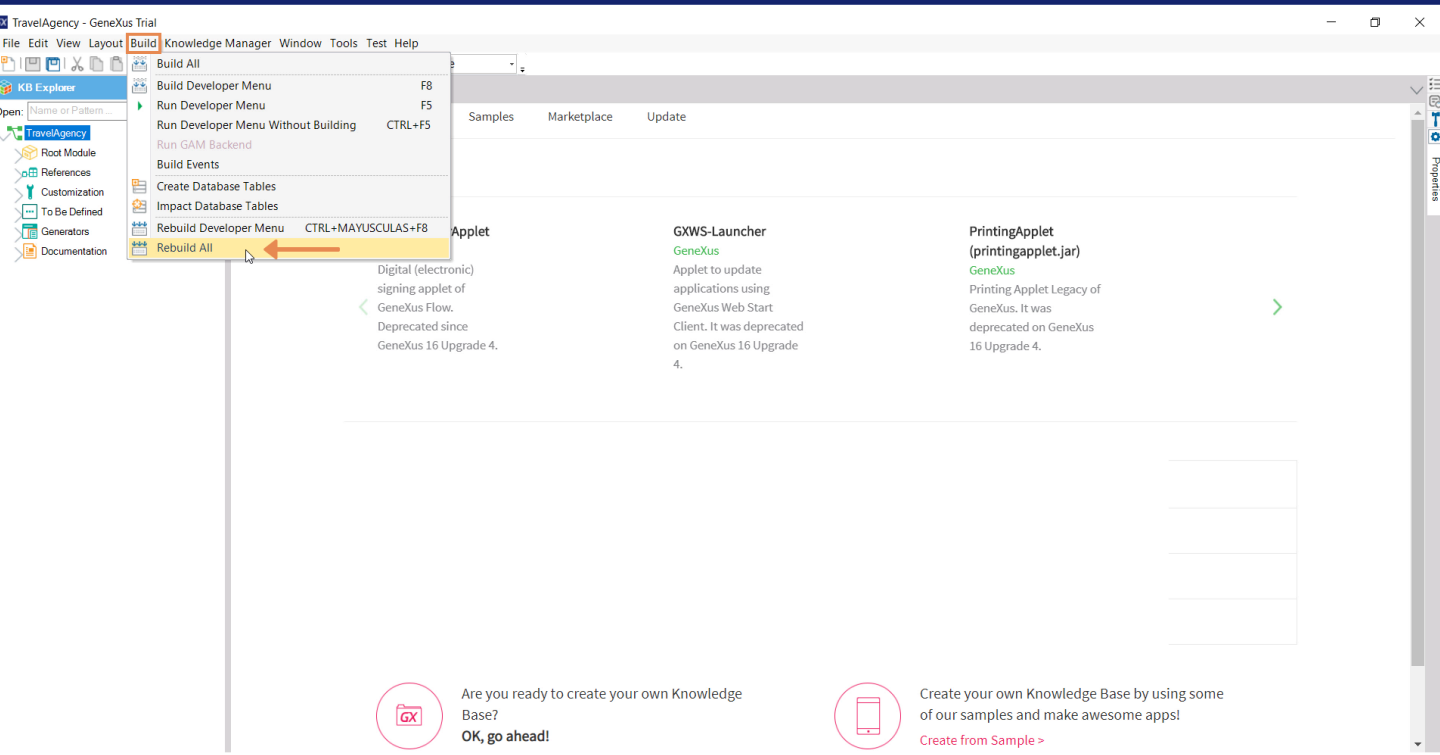


La última propiedad del grupo es ***Validation message display***, que cuenta con los valores *One at a time* (que es el valor por defecto) y cuyo objetivo es mostrar siempre el último mensaje de validación que activa la aplicación,

Validation message display property



y *All at once,* que muestra al mismo tiempo cada mensaje de validación que debería estar en la pantalla.



Como estas propiedades modifican el comportamiento relativo a la interacción en los forms, se deben volver a generar todos los objetos que tienen form para que los cambios sean efectuados. Para ello, debe utilizar la opción **Rebuild All**, que genera absolutamente todos los objetos.

En el siguiente video agregaremos algunas reglas interesantes a las ya conocidas y en otros estudiaremos cómo se evalúan para determinar el orden de ejecución, así como profundizaremos un poco más en los eventos de disparo.

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