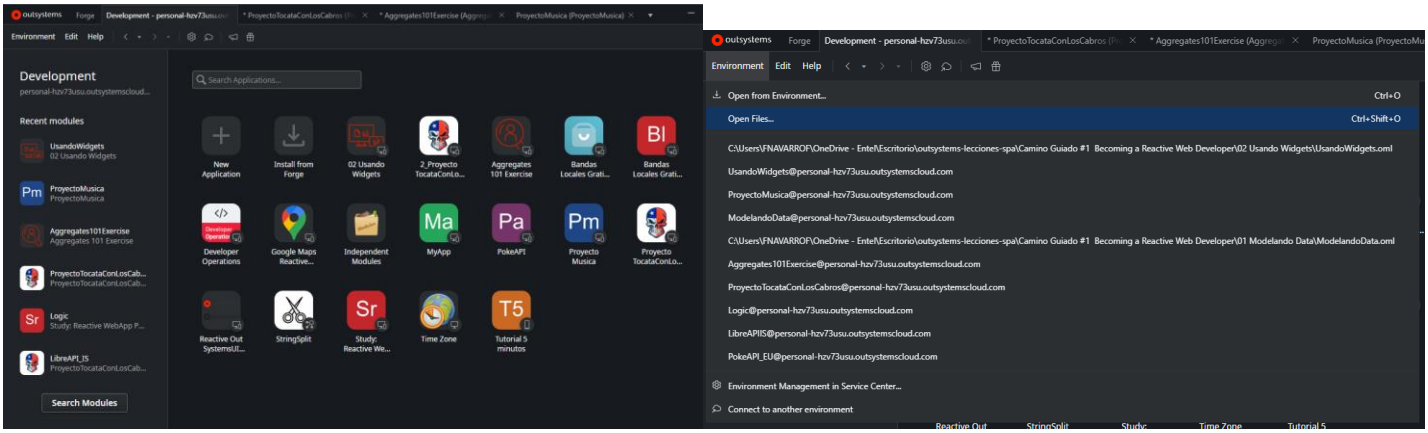
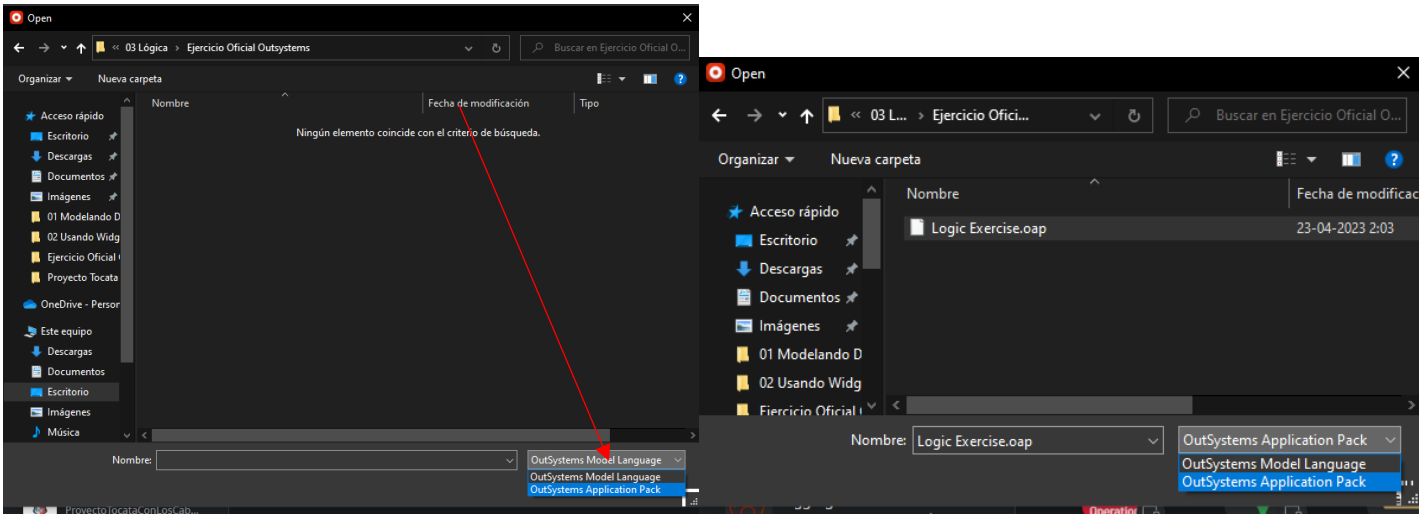
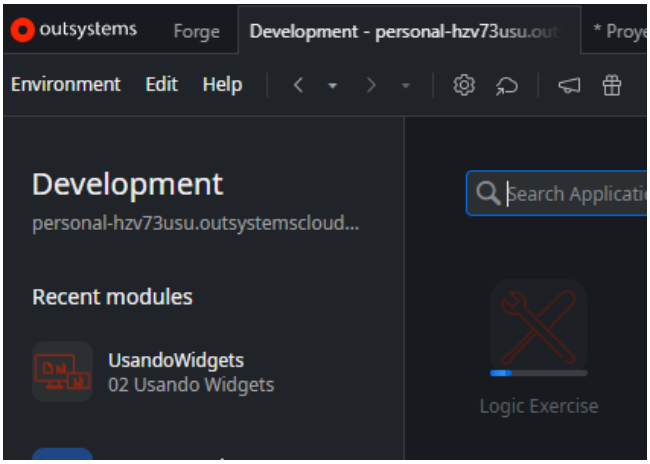
Ahora haremos Acciones, que un botón reconozca los datos mediante una lógica y muestre el resultado.

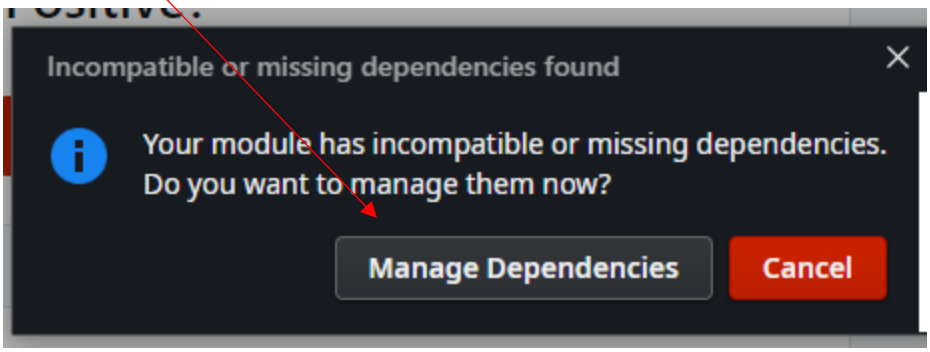




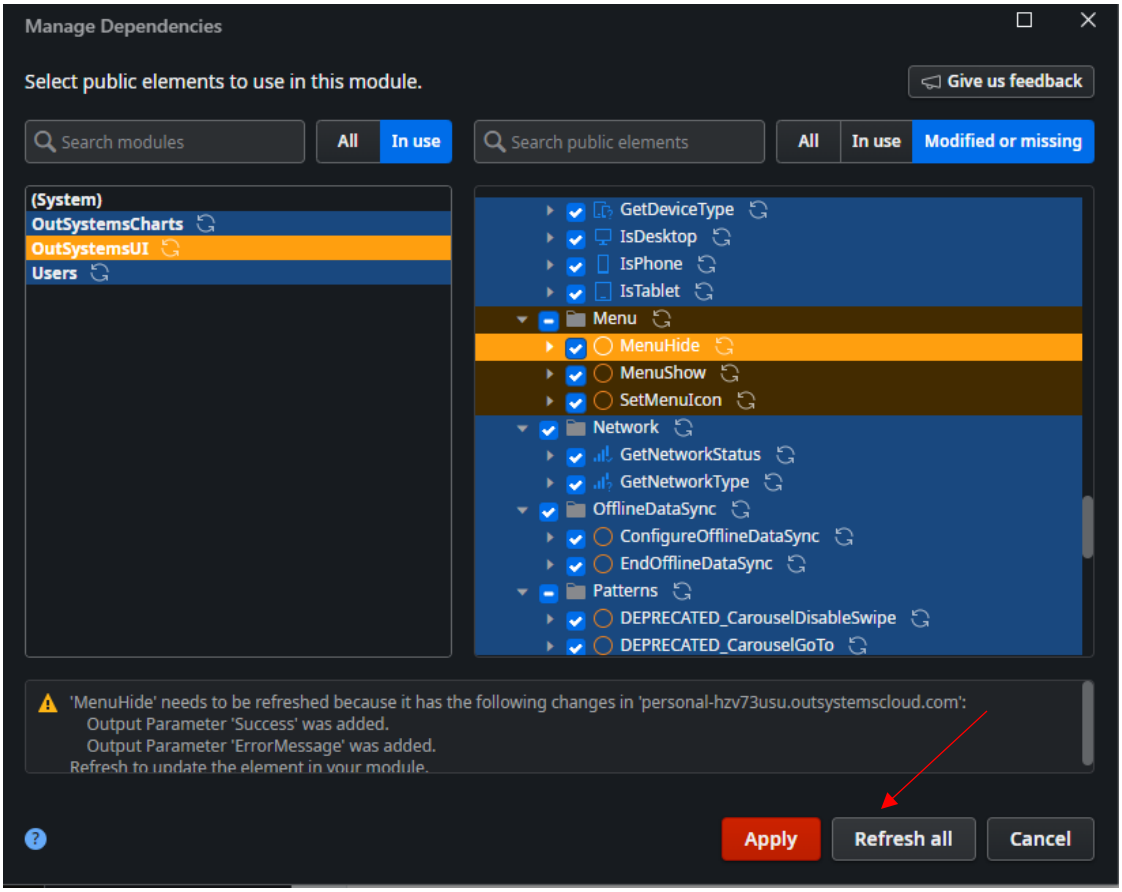
Importamos la aplicación .oap de la carpeta Ejercicio Oficial Outsystems. No se te olvide cambiar la extensión de la aplicación empaquetada para que reconozca el archivo “Logic Exercise.oap”



Una vez instalada la abrimos.

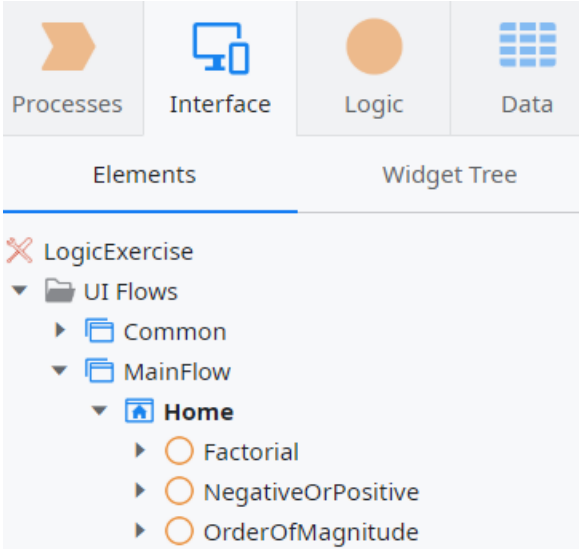


Te dirá que tienes dependencias desactualizadas, porque cuando se hizo el tutorial, la versión de outsystem y sus módulos eran antiguos y ahora están en desuso. Elije Manage Dependencies.

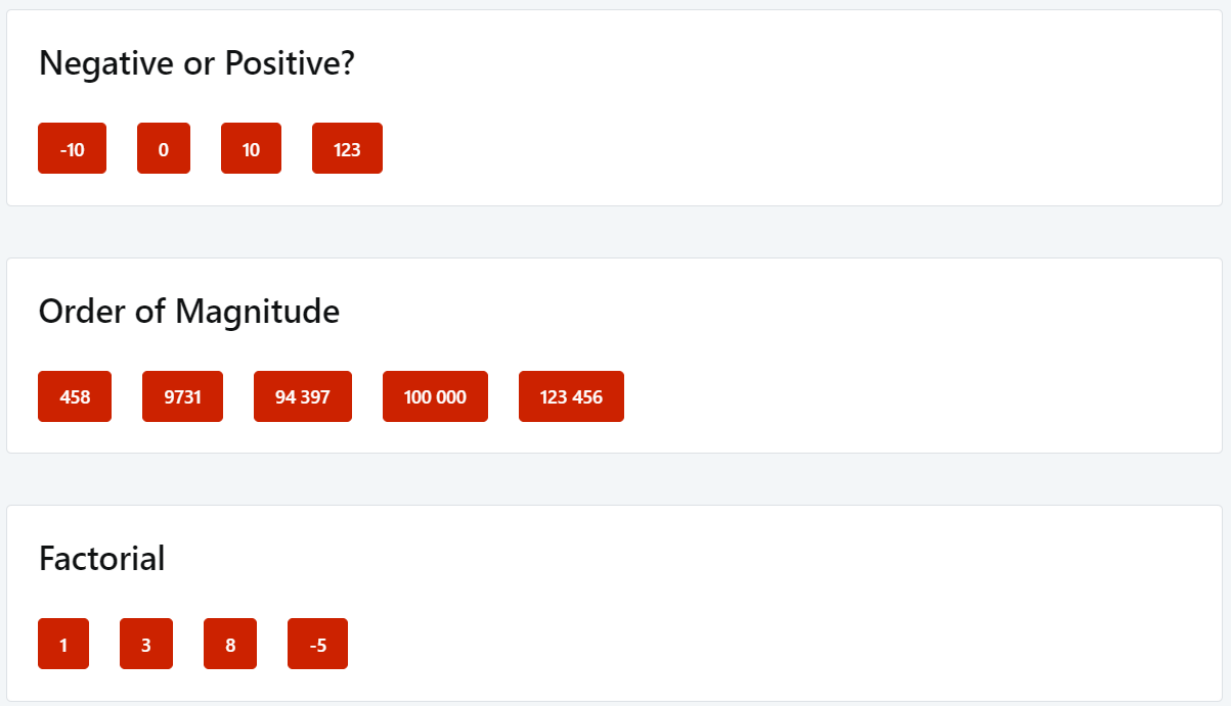


Tienes que seleccionar Refresh All, y luego Apply.

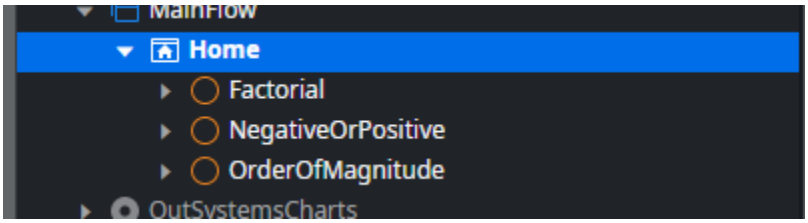
Si no lo haces dará error, y con errores no puedes publicar la App.



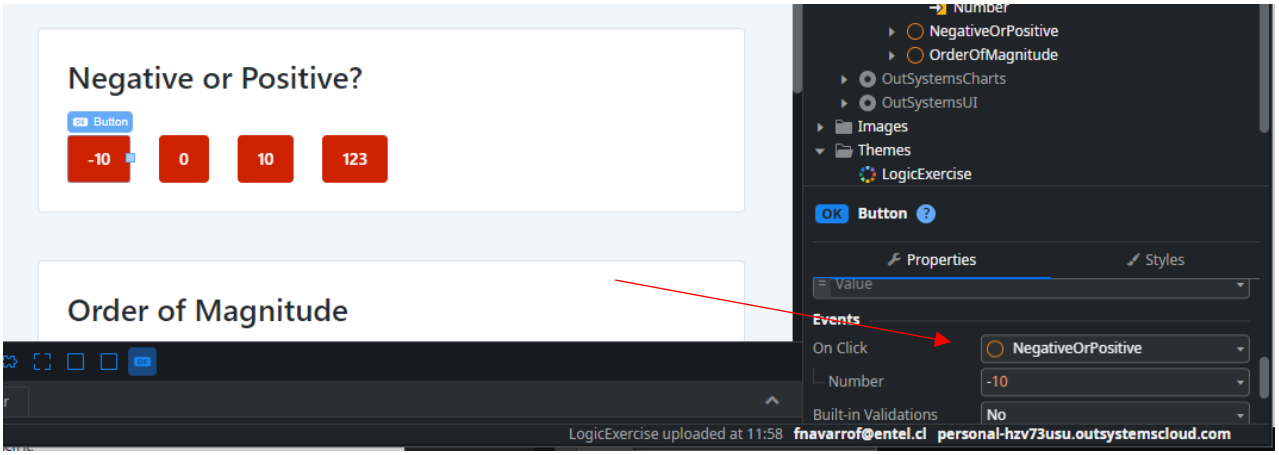
Vamos a la pantalla Home.



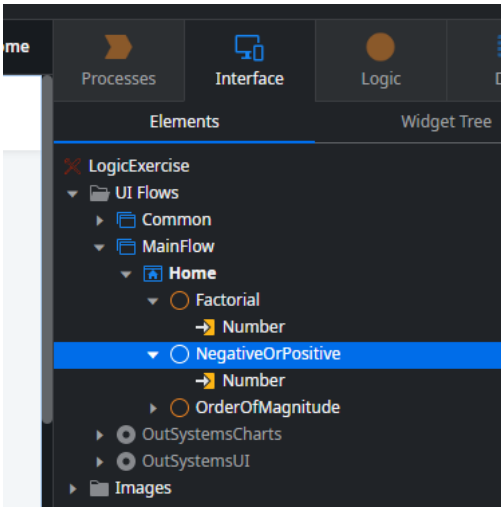
La pantalla tiene una sección de Negativos o Positivos; Orden de Magnitud; y Factorial.



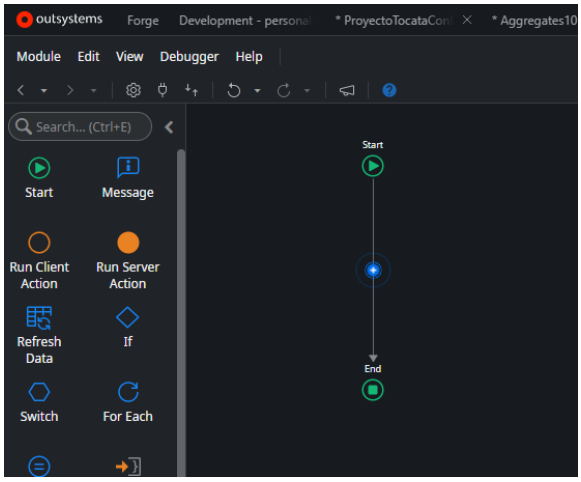
Y ya están creadas las Client Actions.



Y cada botón ya está configurado con un OnClick que dirige al Client Action. En este caso, el -10 dirige al Client Action de NegativeOrPositive y seteado con su valor -10.



Lo que tenemos que hacer ahora es hacer doble click en esa Client Action.

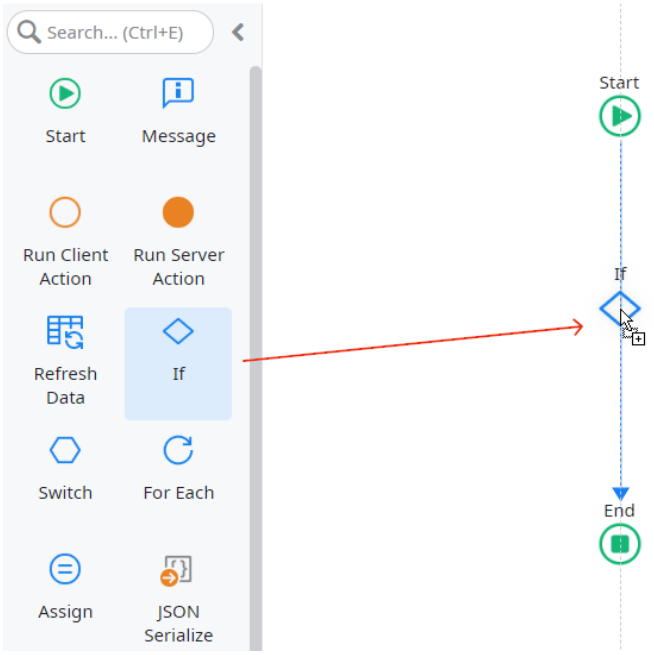


Y crear una lógica que me de un mensaje informativo así como “Hey, este número es negativo”.

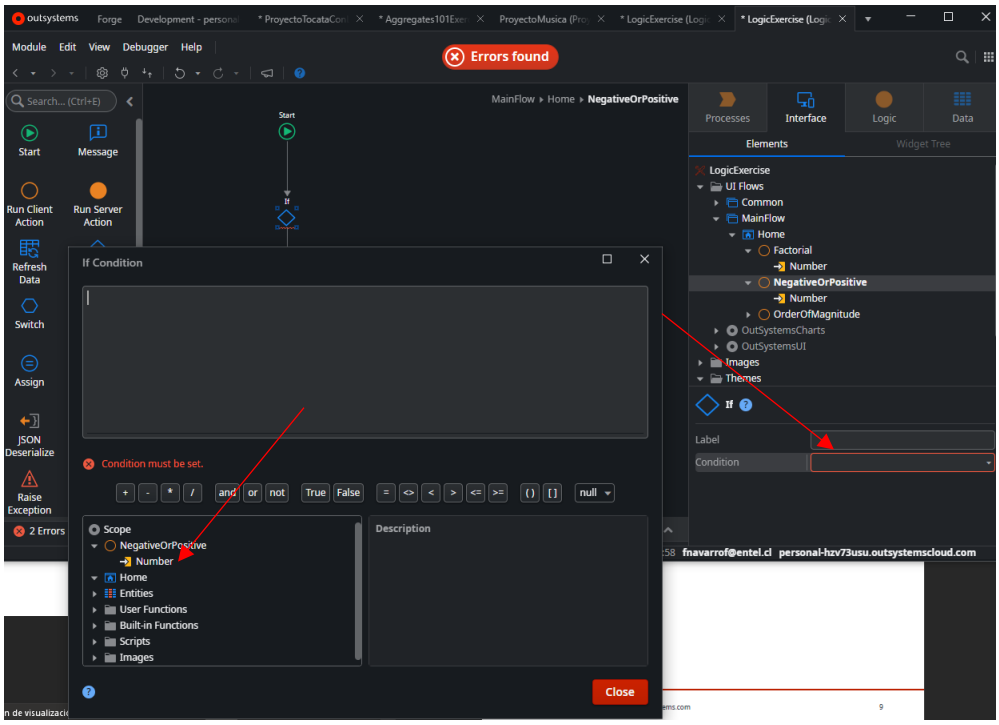
¿Listo? Empezamos.

POSITIVO O NEGATIVO – LÓGICA

Ocupando If

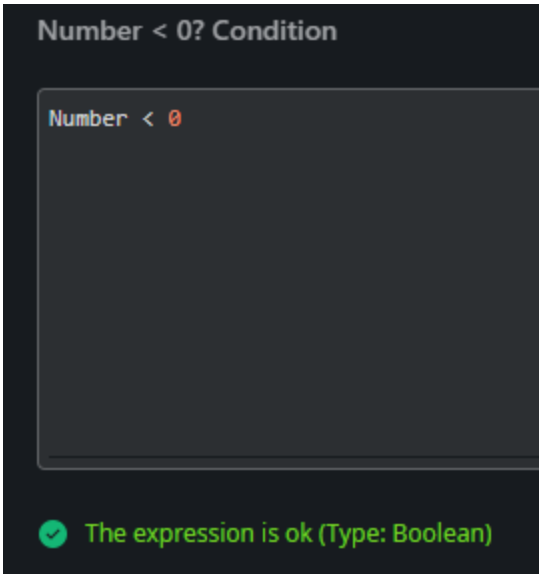


Arrastramos un If en medio del flujo de la lógica.

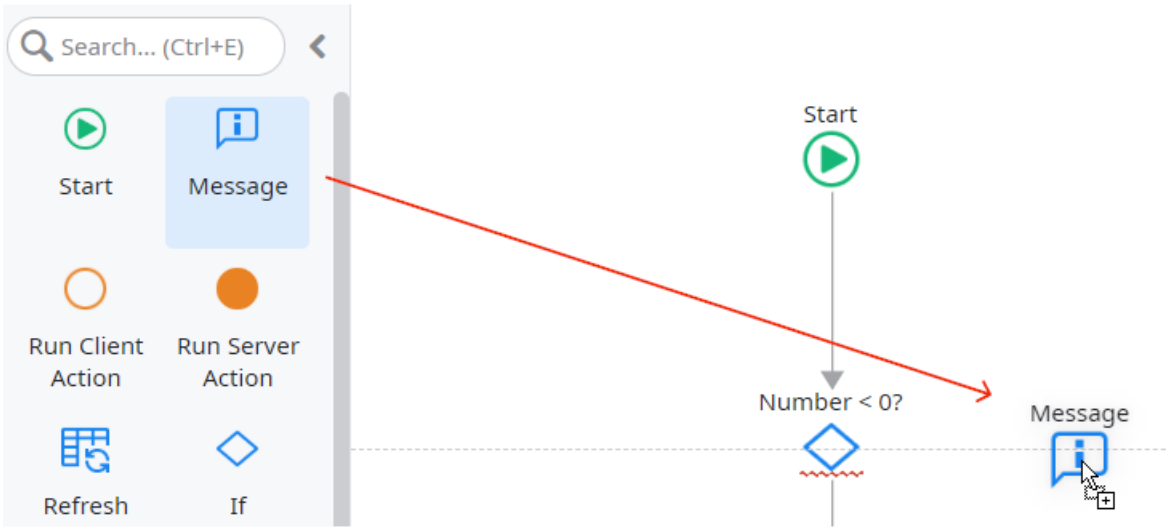


Doble Click a la condición y seteamos que el número sea menor a cero.

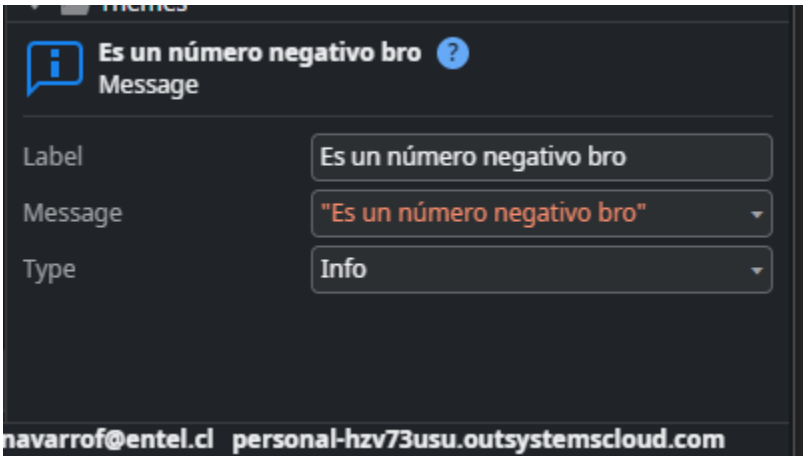
Podemos hacer doble click en la variable input Number, para que se escriba y luego escribir < 0



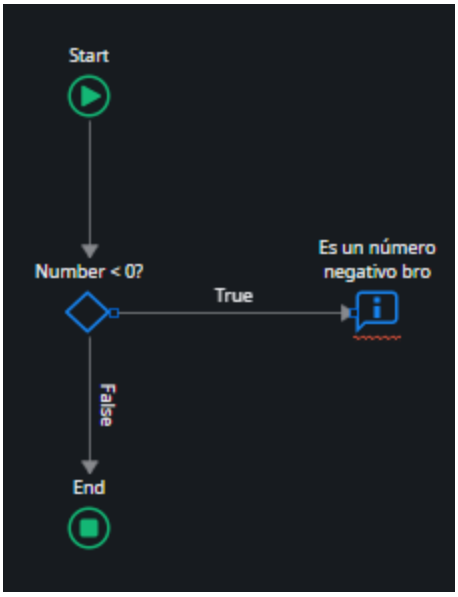
Close.



Añade un mensaje a la derecha del flujo. Y escríbele “Es un número negativo bro”, en Message y entrecomillas para que se detecte como String.

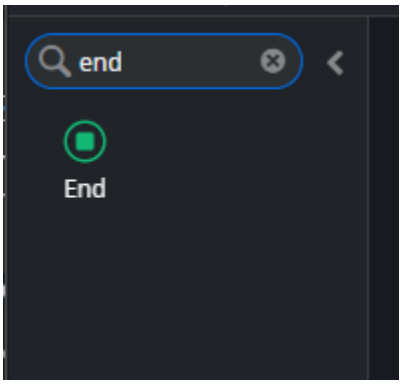


Ahora vamos a crear una rama True. ¿Un condicional puede tener una tercera rama? No, solo dos, True o False.

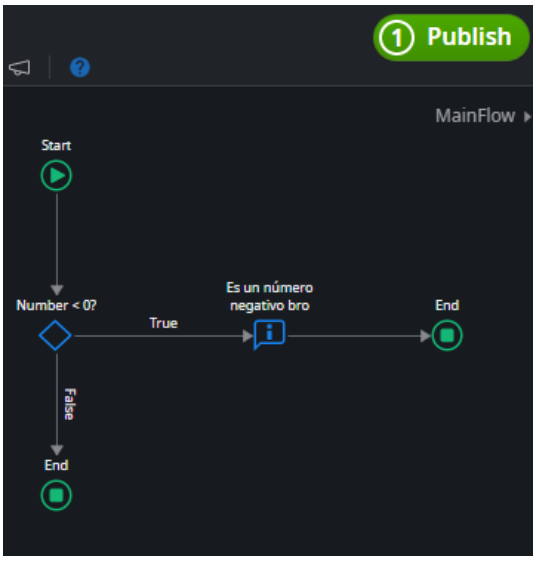


Arrastramos desde el condicional al Message, para crear la rama true.

¿Por qué sigue dando error? Porque falta un End para terminar la rama true, o sino sigue abierta sin final.

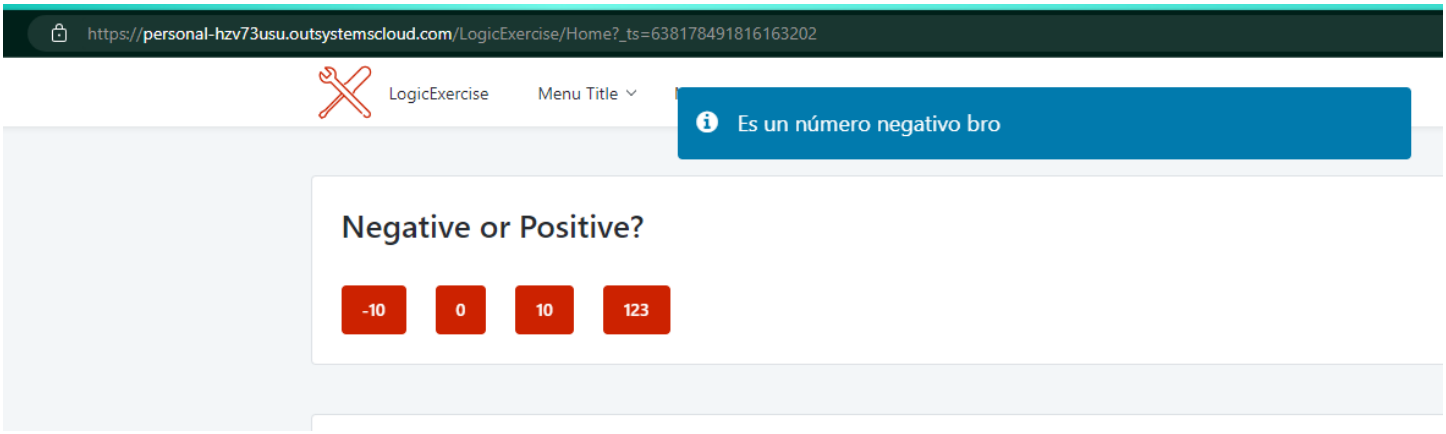


Buscamos su End y arrastramos



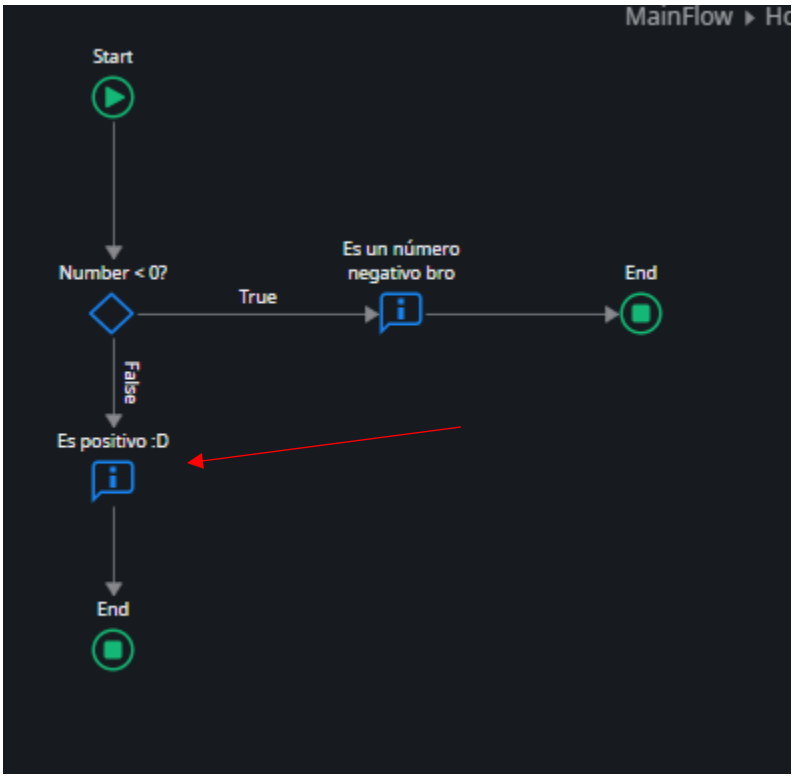
Y cerramos el circuito de la lógica.

Publicamos y probamos.

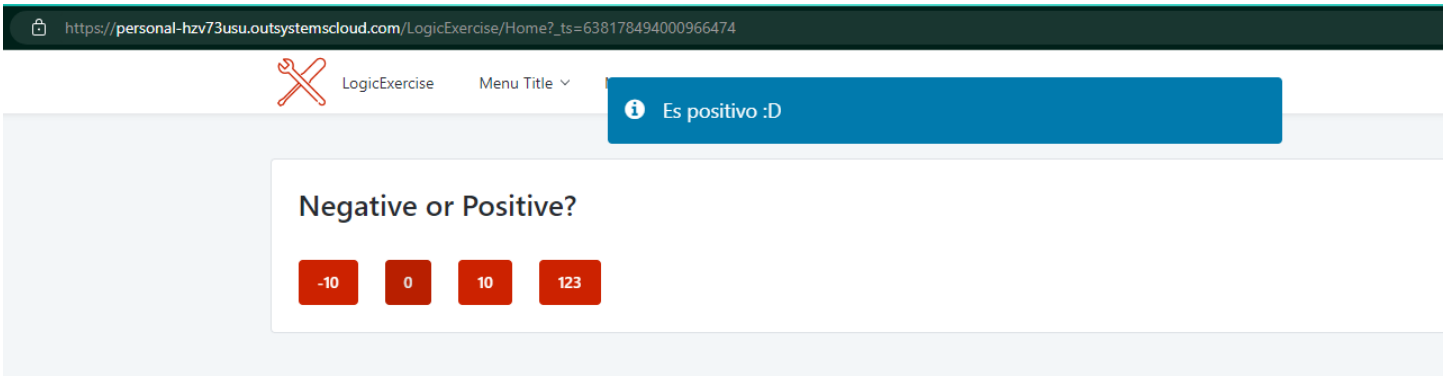


Funciona, solo para el primero. Para los demás números la Lógica se va a un End sin mensaje.

Le añadimos el mensaje positivo en el circuito.

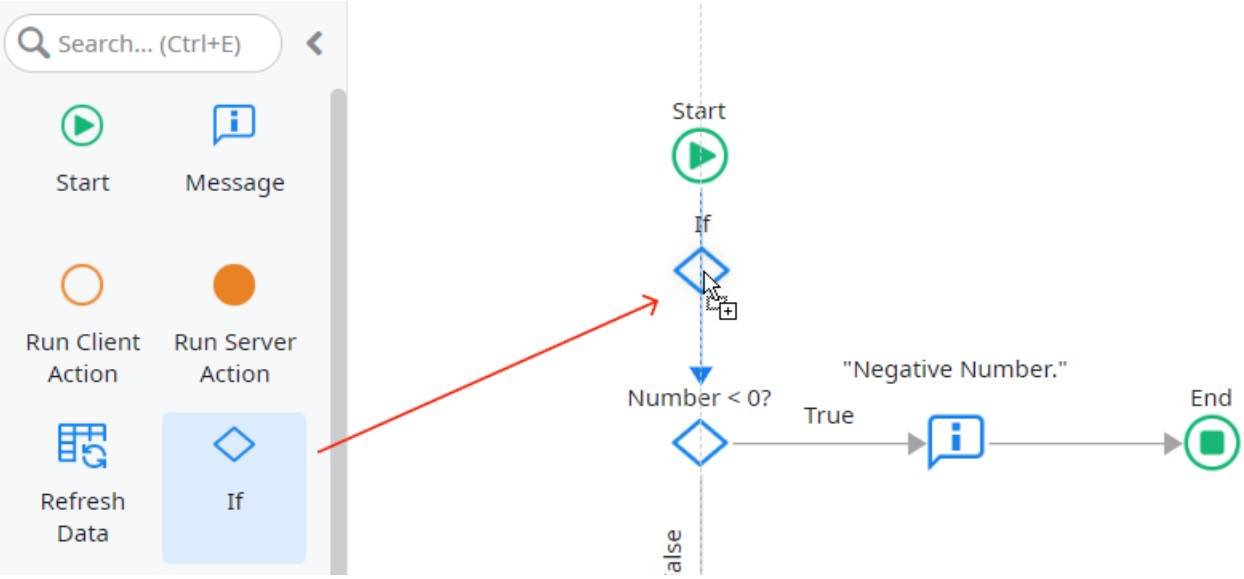


Publicamos y probamos.

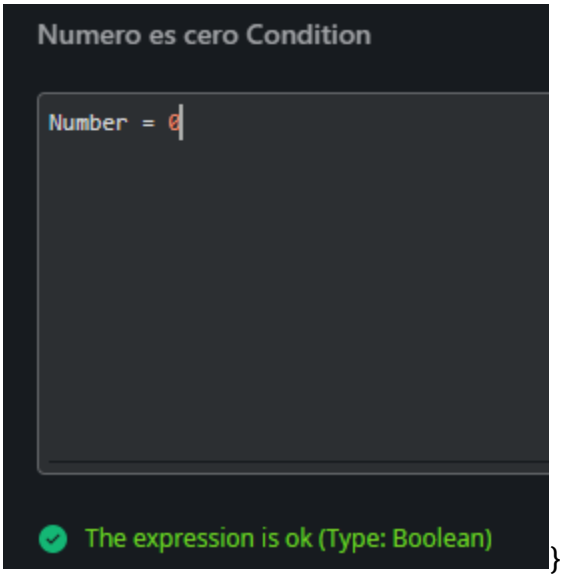


Ahora hasta los demás dan positivo. Hasta el cero da positivo.

Arreglaremos ese bug.

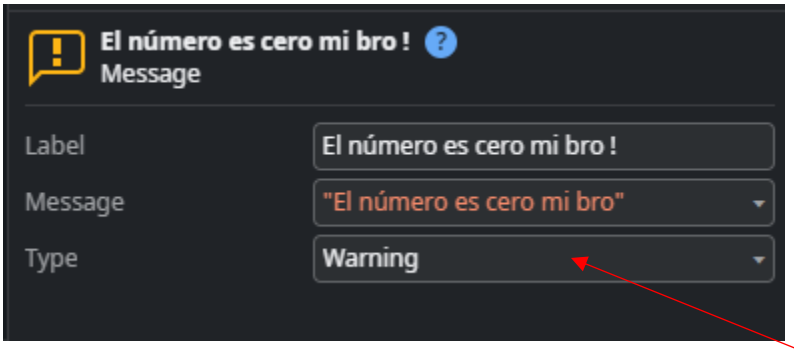


Podemos añadir un condicional antes que inicie el condicional para detectar si es negativo o no.

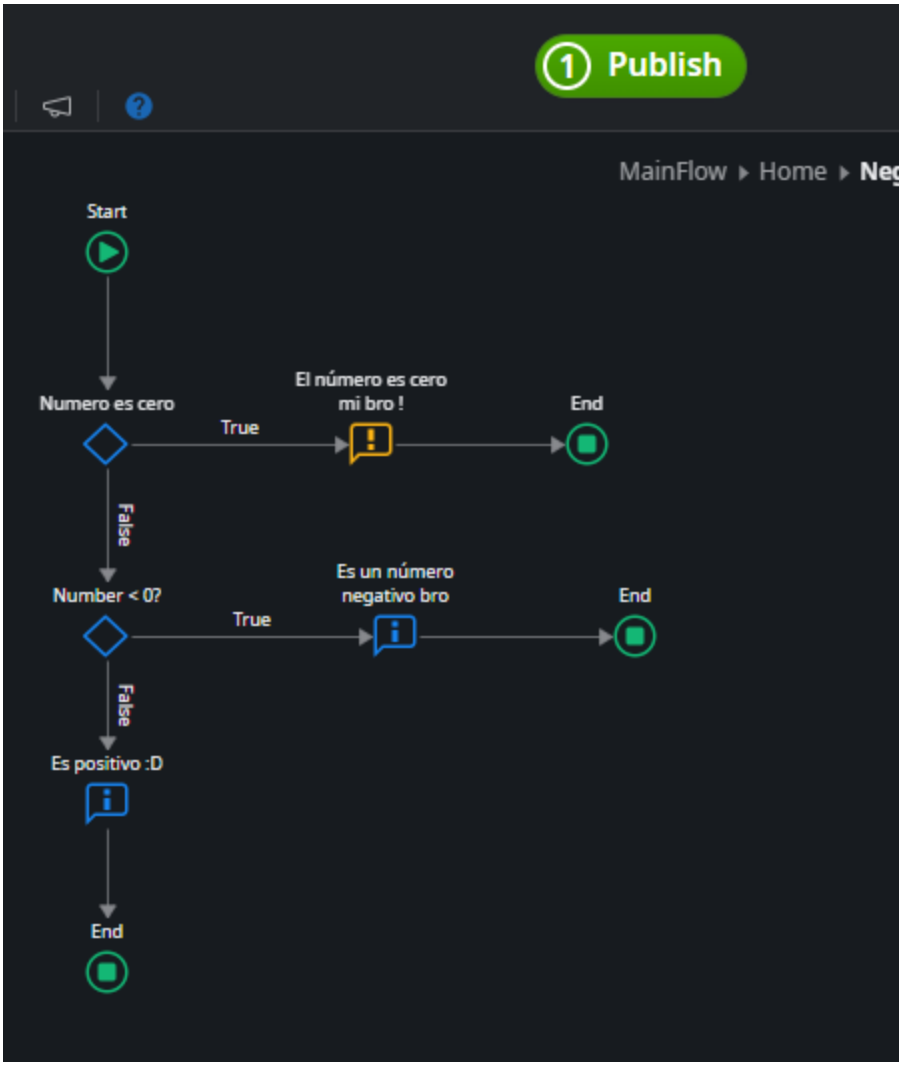


}

Seteamos con la condición Number = 0.

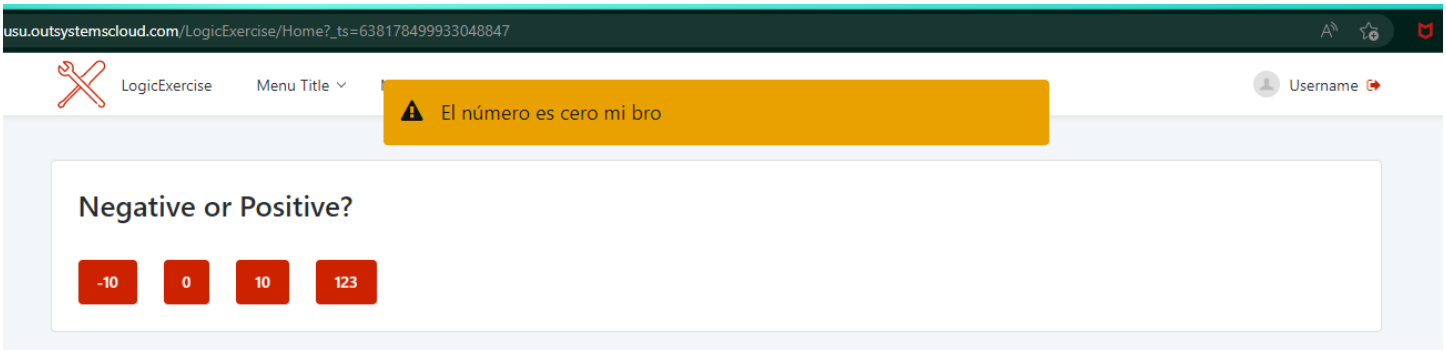


Ocuparemos el tipo Warning



Añadimos también el mensaje y lo cerramos con End.

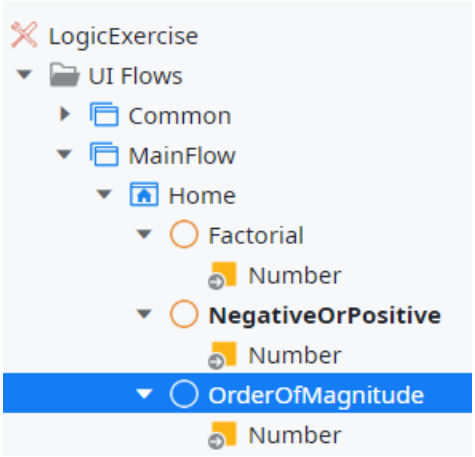
Publicamos y probamos.



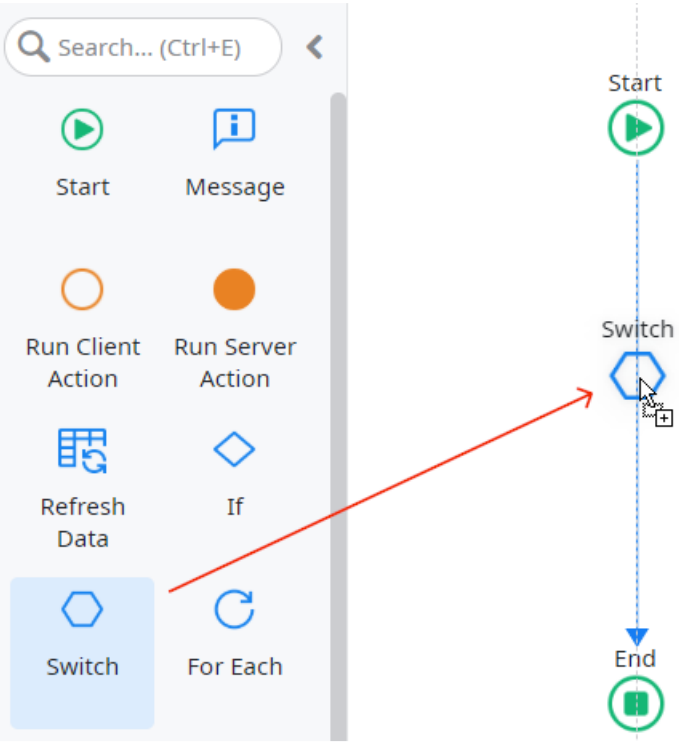
Listo.

ORDEN DE MAGNITUD – LÓGICA

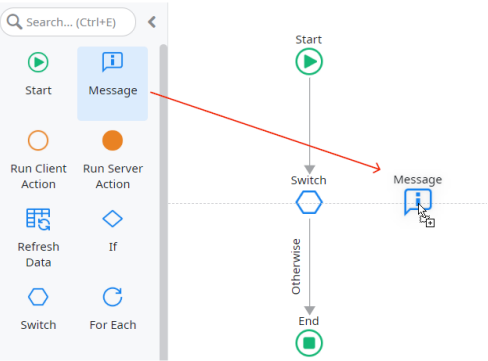
Ocupando el Switch



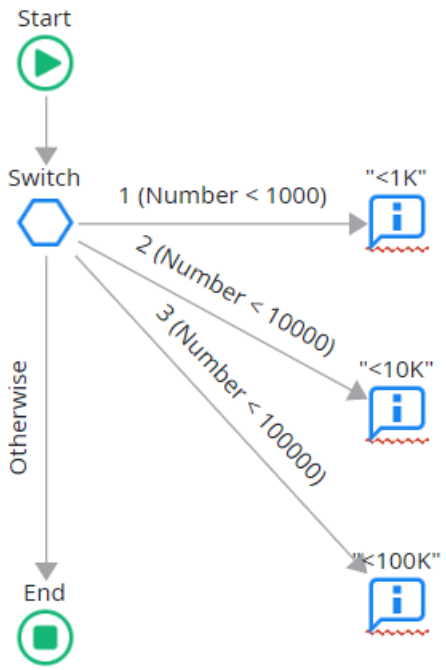
Abrimos la acción OrderOfMagnitude.



Añadimos un Switch

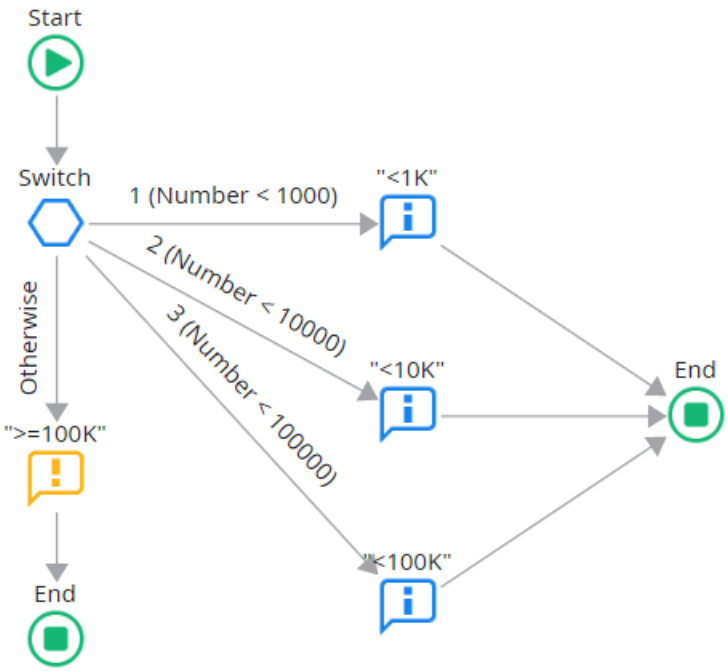


Añadimos un mensaje a la derecha y al conectarlo le ponemos la condición “Number < 1000”



Y con Switch podemos hacer más de dos ramas. Configuramos 4 condicionales.

- Number < 1000
- Number < 10000
- Number < 100000
- Otherwise



Añade un End a los primeros 3 condicionales. Y al Otherwise añade un mensaje warning que diga “>=100K”.

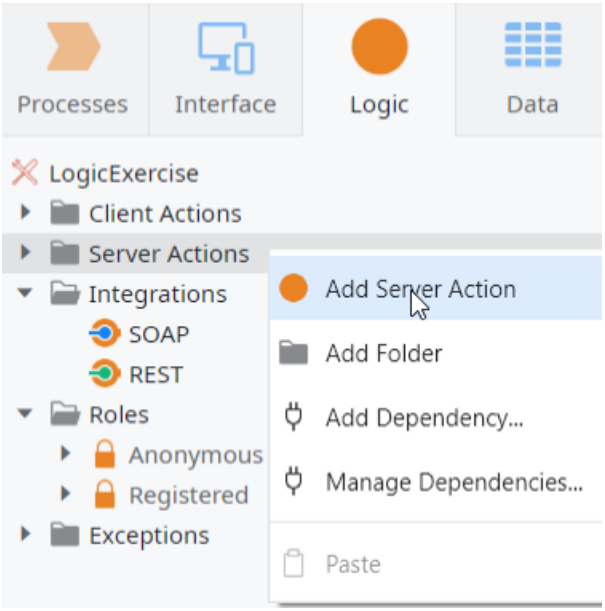
Publica y prueba.

FACTORIAL

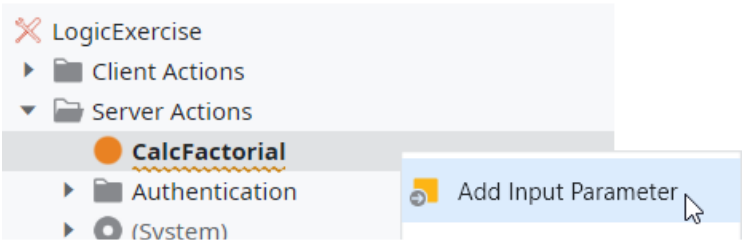
Usando Server Action y Assign

```
4! = 4 x 3 x 2 x 1 = 24
7! = 7 x 6 x 5 x 4 x 3 x 2 x 1 = 5040
```

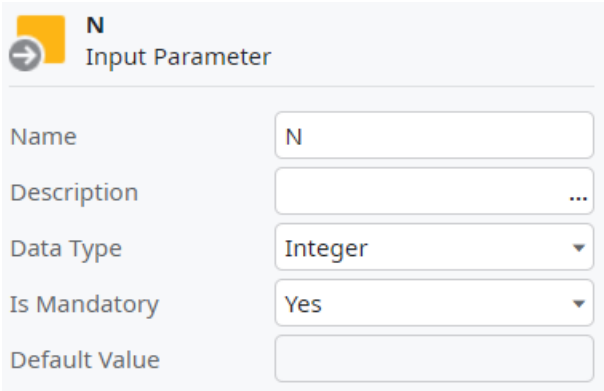
Necesitamos calcular el factorial de los números, como el ejemplo de arriba. 4 su factorial 4! Da como resultado 24.



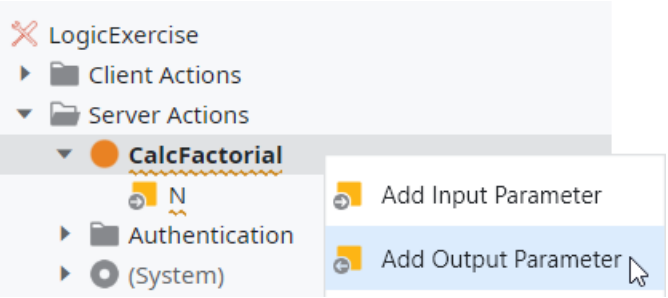
Vamos a la capa de lógica y en Server Action creamos una Server Action llamada CalcFactorial.



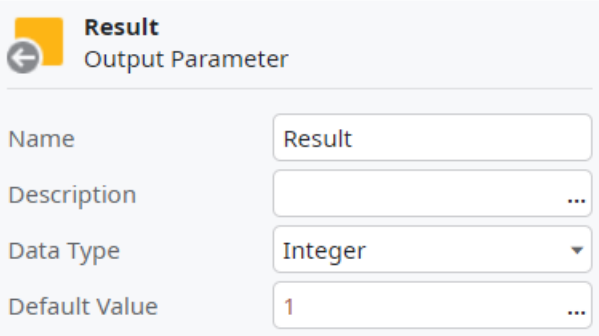
Dentro de esa acción creamos una variable Input Parameter.



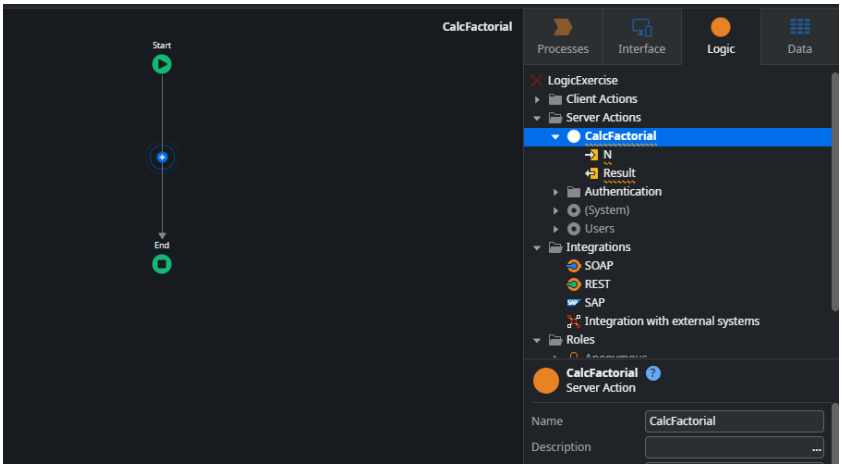
La llamamos N y ponemos Data Type: Integer y que sea Mandatoria.



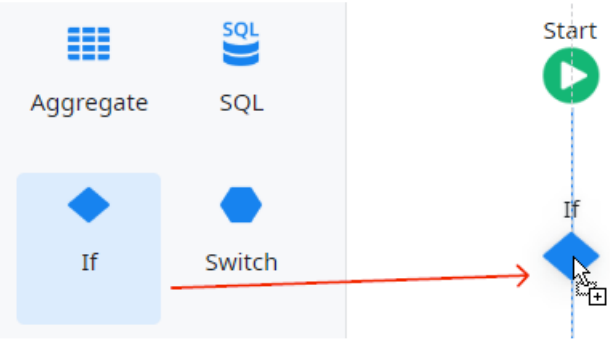
También hacemos una variable de salida, un Output Parameter.



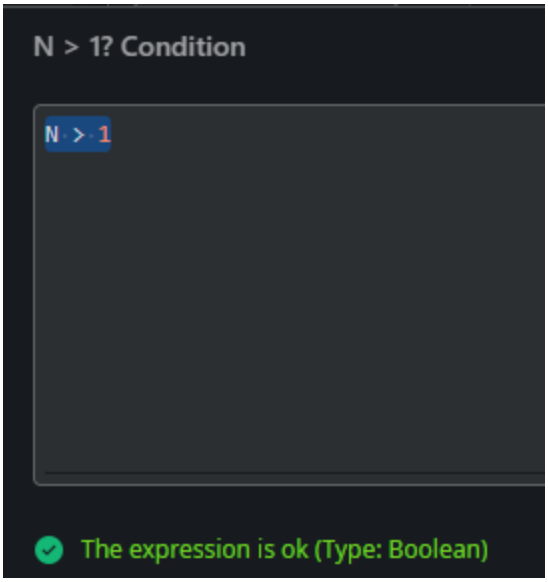
Llamada Result, tipo Integer y con valor por defecto de 1.



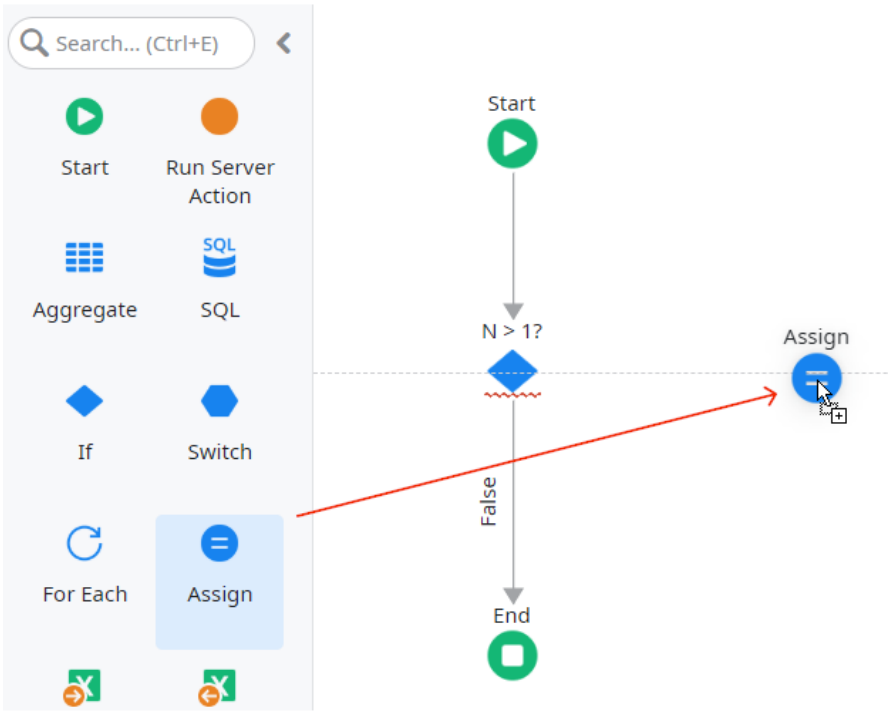
Vamos a la Lógica del Método CalFactorial.



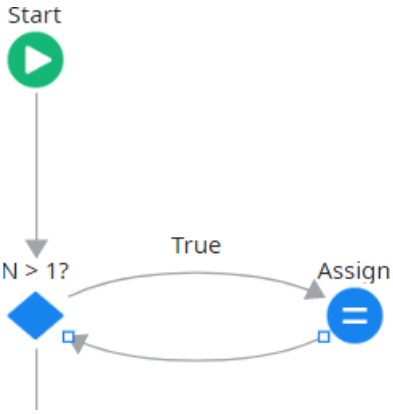
Añadimos un If



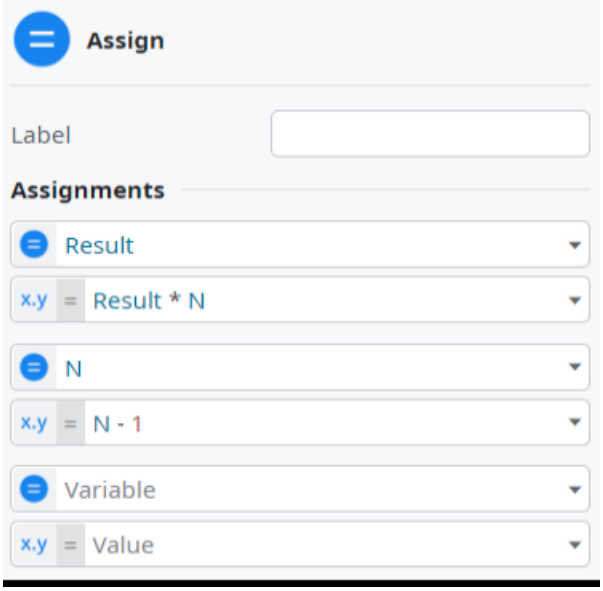
Aplicamos condición N > 1



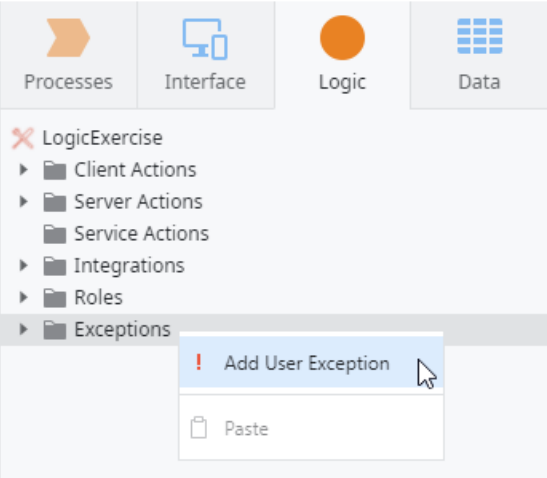
Añade un Assign



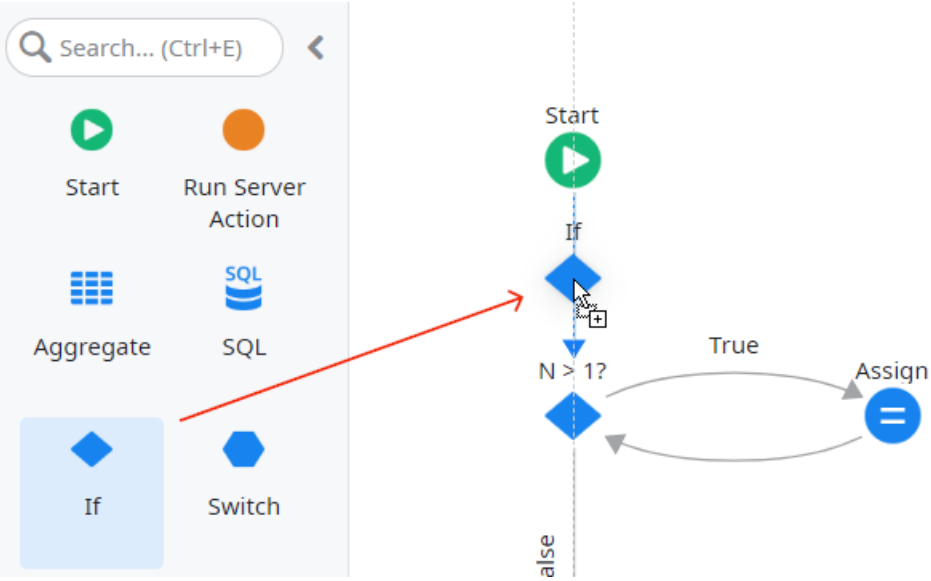
Y añade conectores como la imagen para hacer un bucle.



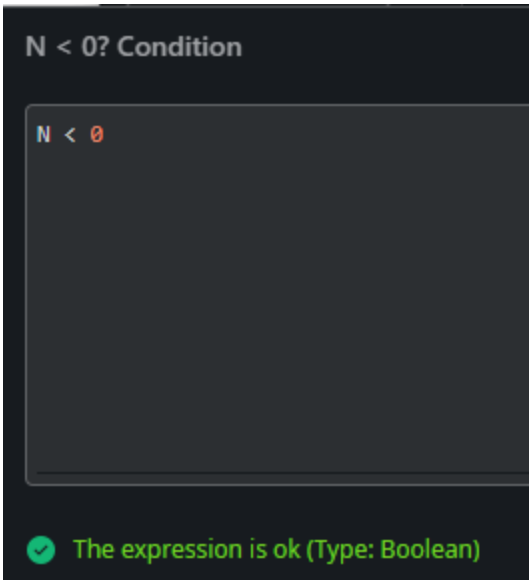
Y configura el Assign con esos parámetros.



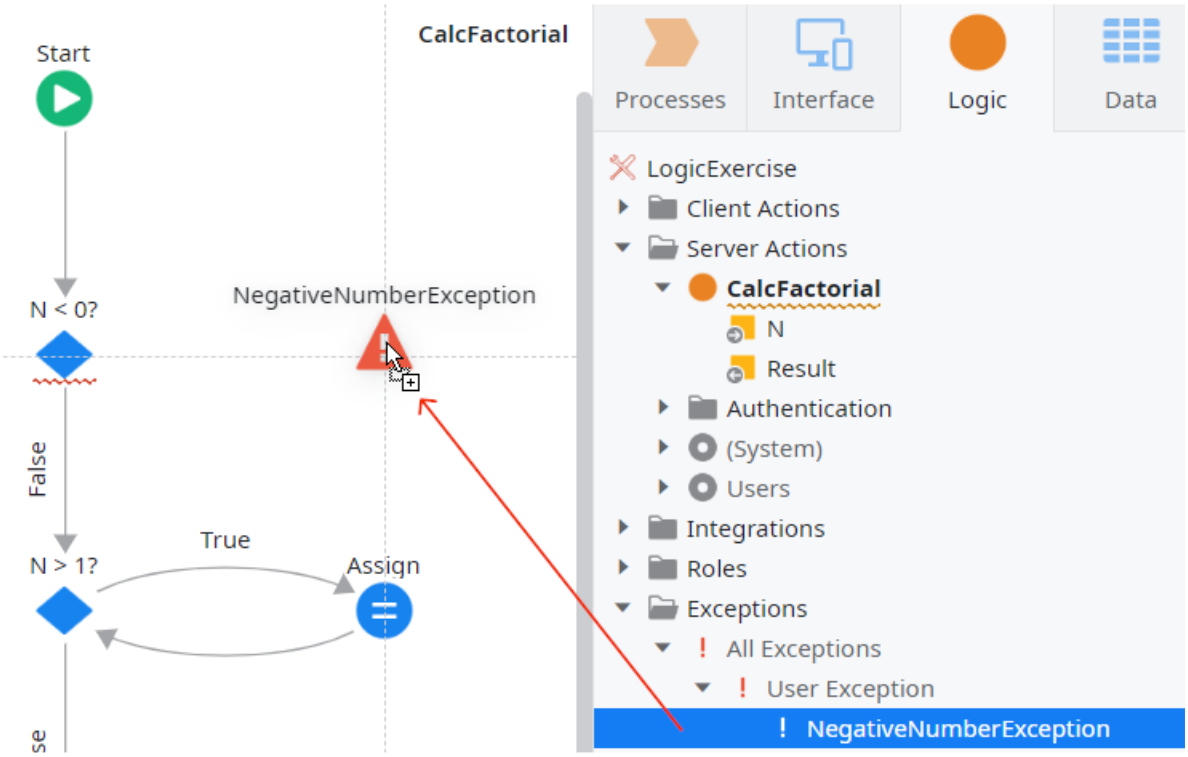
Anda a la carpeta Exception y añade una nueva excepción llamada NegativeNumberException



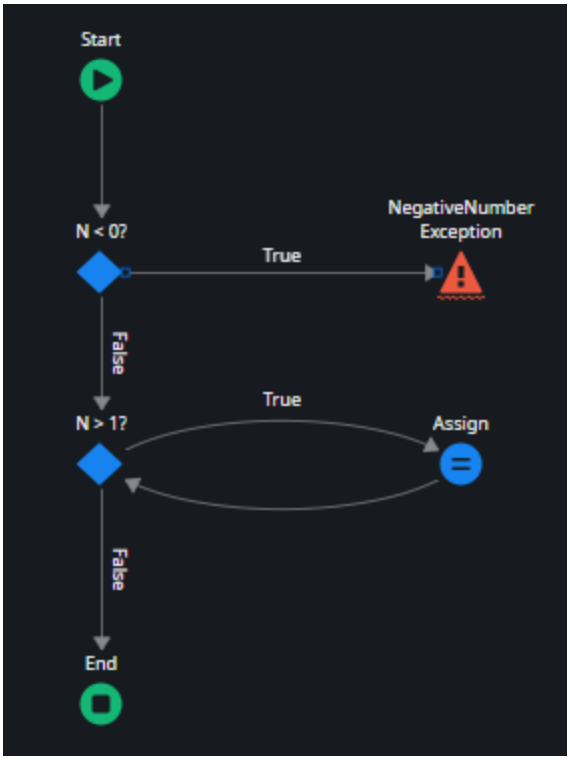
Añade otro If como al imagen.



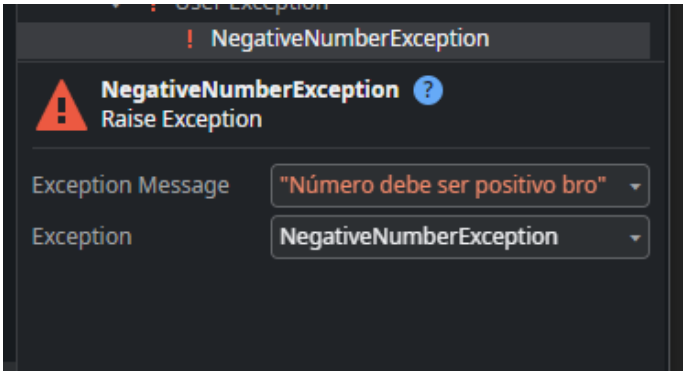
Setea esa condición.



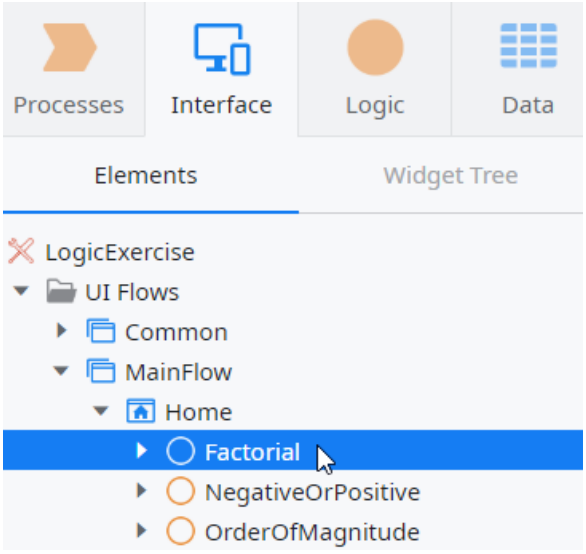
Arrastra la excepción a la derecha del If recién creado.



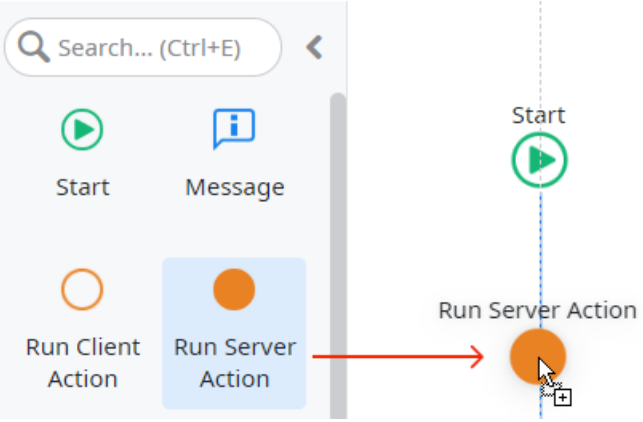
Crea la rama True del condicional a NegativeNumberException



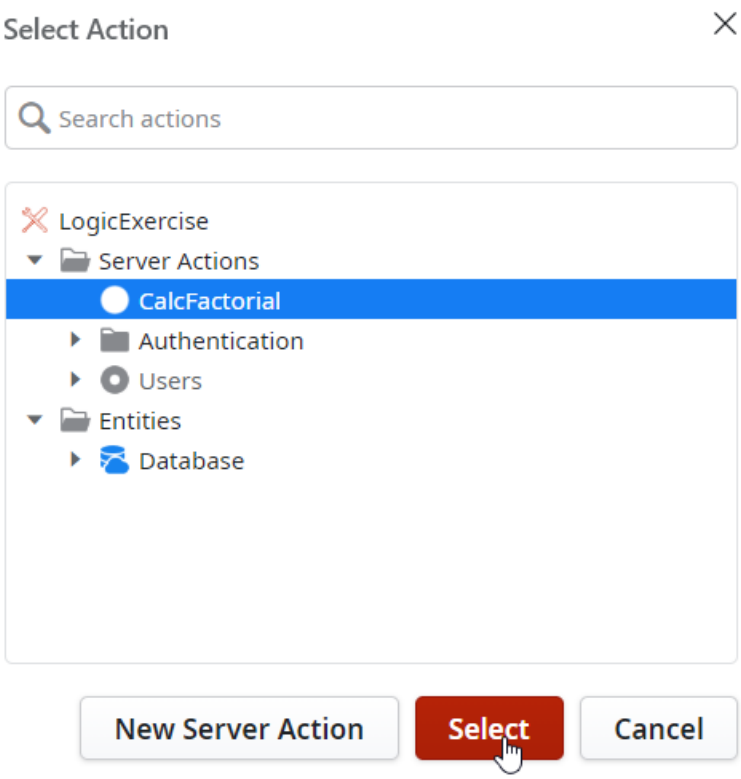
Configura eso en la excepción.



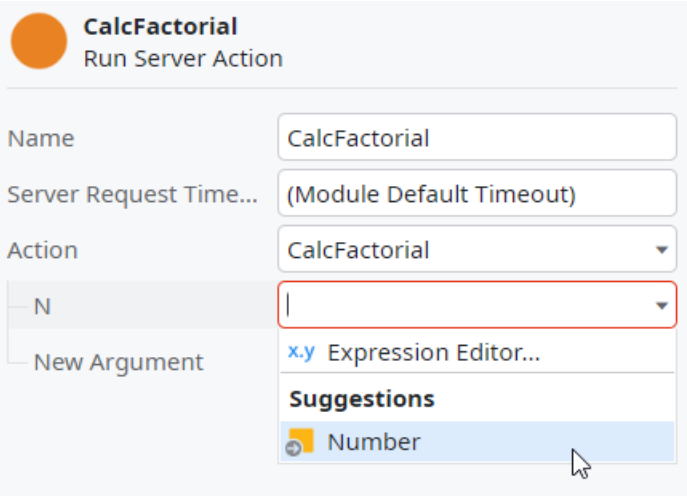
Ahora vamos a Interface → Home → Factorial, y modificamos esa Screen Action



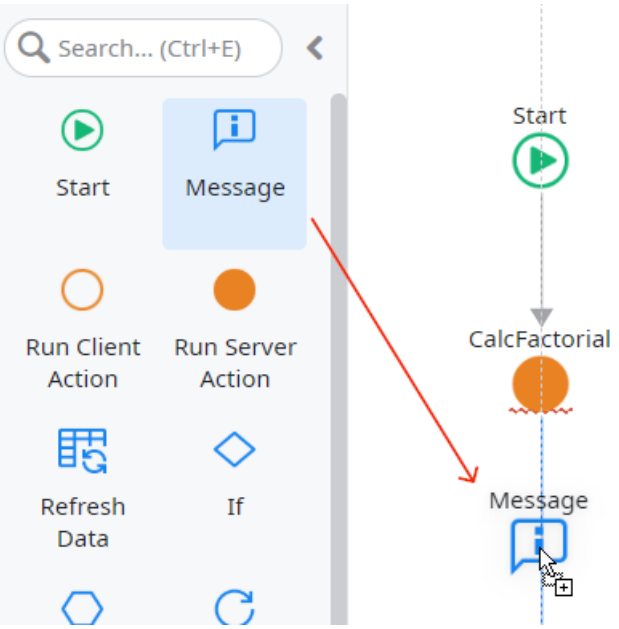
Arrastramos Run Server Action y por supuesto llamaremos la Server Action que acabamos de crear, CalcFactorial.



Seleccionamos la Server Action

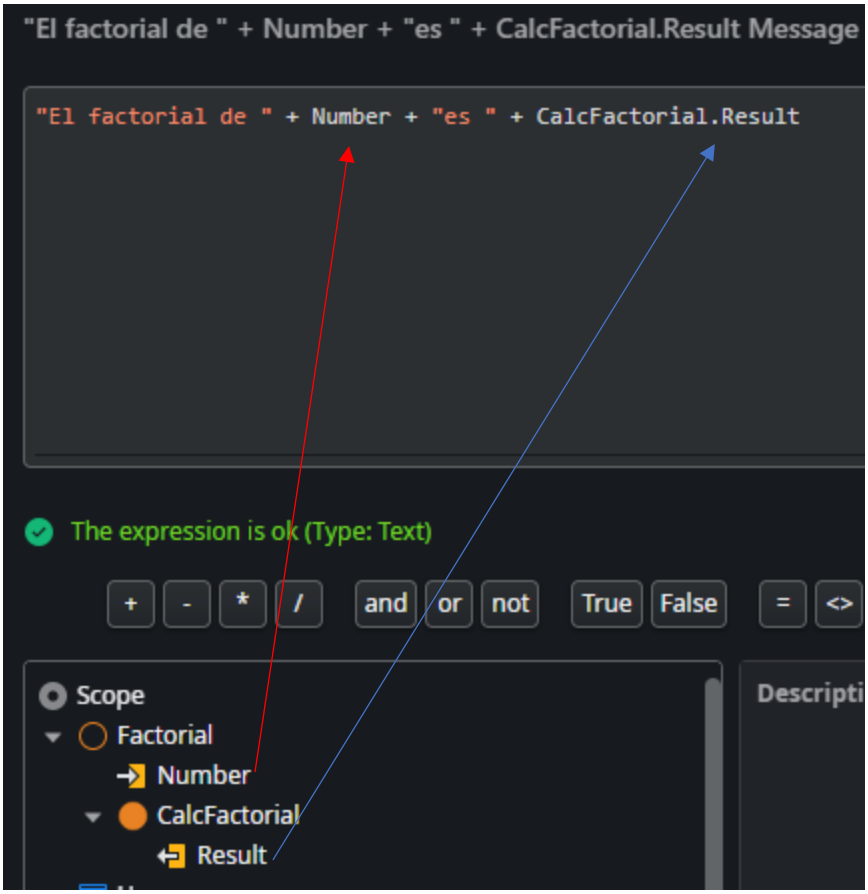


En el método de CalcFactorial tenemos que configurar N (Como llamamos el Input Parameter del método server action CalcFactorial) con Number (como llamamos el input parameter del método screen action de la pantalla Home).



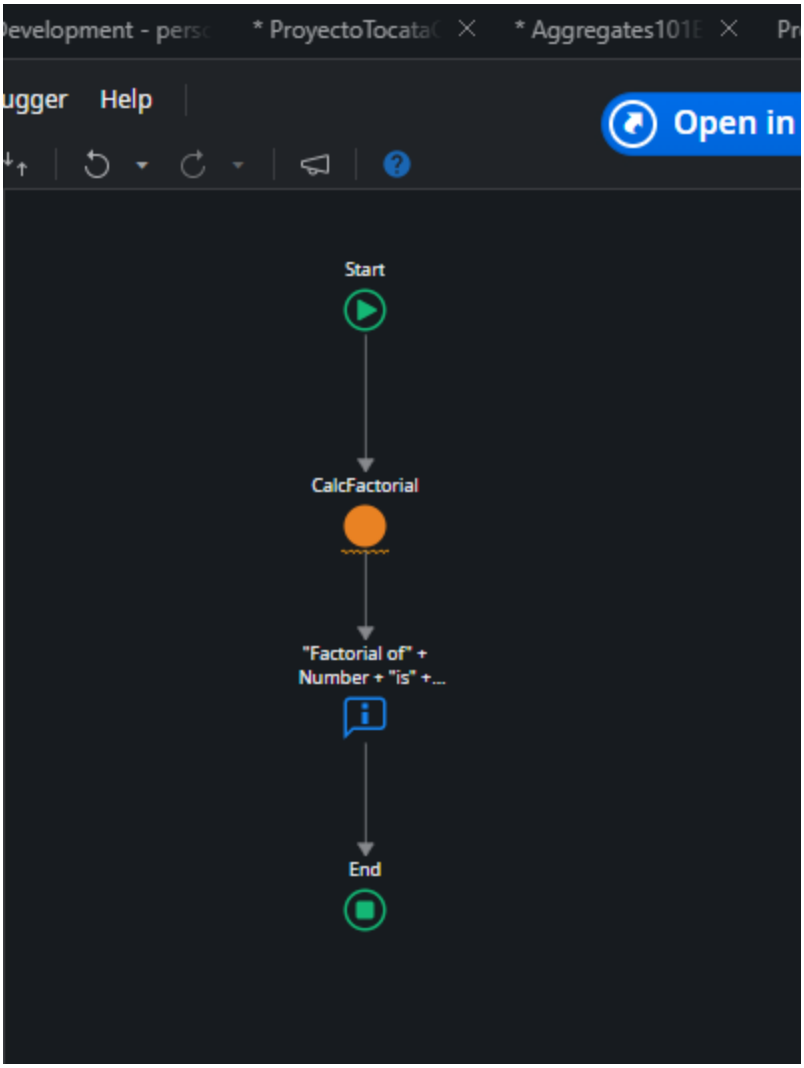
Arrastra un mensaje y que otorgue el mensaje:

```
"Factorial of " + Number + " is " + CalcFactorial.Result
```

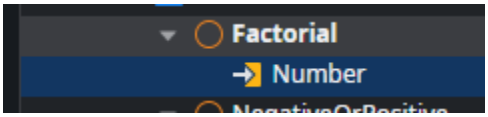


Publica y prueba.

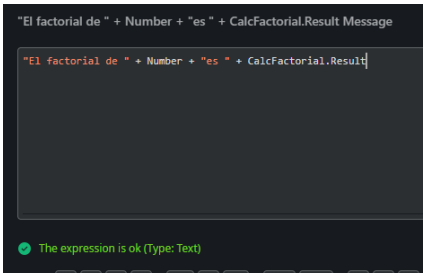
Explicación del método:



El método de lógica de la Screen llamado Factorial tiene este flujo. Y dentro de este flujo llama a una acción del servidor (que corre en el servidor).

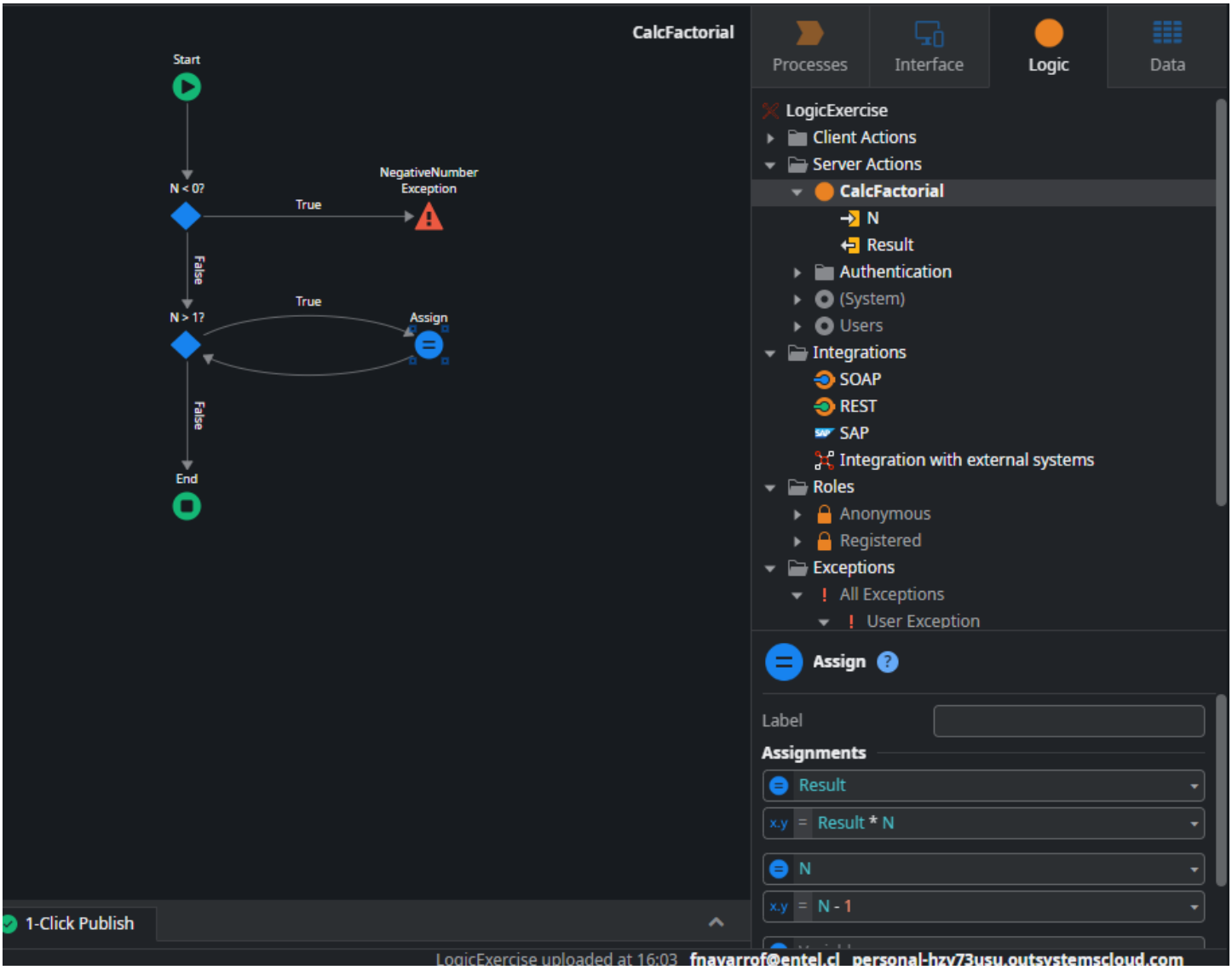


Y este método de servidor va a recibir la variable Number que tenga esa Screen. No tiene Output porque las Screen Action NO TIENEN VARIABLES DE SALIDA (OUTPUT PARAMETER). *No tiene utilidad exportar el dato en una variable si estamos en la misma pantalla. Estando en la misma pantalla podemos llamar el dato con un message por ejemplo.*



Podemos llamar al servidor eso sí, para que trabaje el dato.

Vamos a la lógica del método hecho en el servidor, método llamado CalcFactorial que está en la pestaña Logic.



Note que el exception no requiere End.

Y aquí lo más complicado es como configurar el Assign.

N recibe el Number de la Screen.

Si N > 1 entra en el bucle.

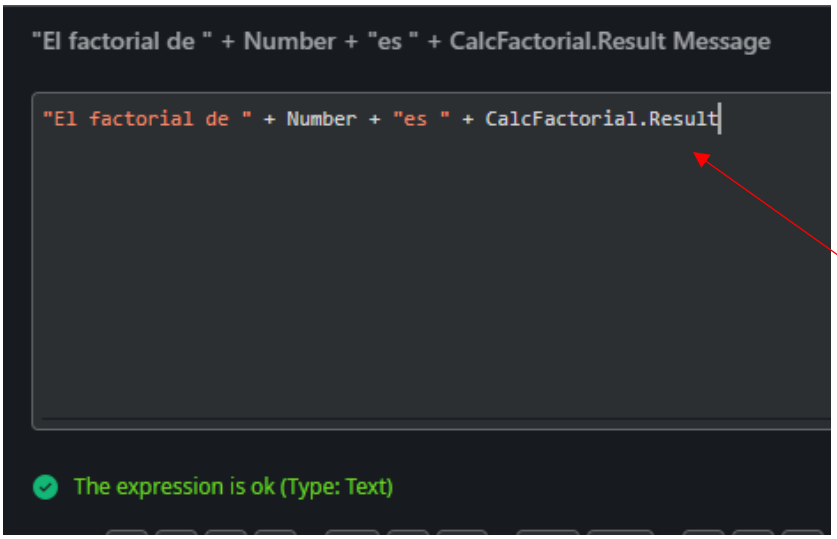
Result configurado con un 1 por defecto.

Result * Number y N se le resta 1➔ Por ejemplo, 3 → 1 * 3 y 3-1 → Result: 3 y N: 2 →

Vuelve a entrar en el bucle → 3*2 y 2-1 → Ahora Result vale 6 y N vale 1 →

No vuelve a entrar en el bucle, porque el condicional N>1 lo frena, hasta ahí llega el bucle y se va al End del método.

Terminó resultando que Result es 6, y este al llamarlo en la Screen se llama CalcFactorial.Result.



Eso es todo por hoy.

:D