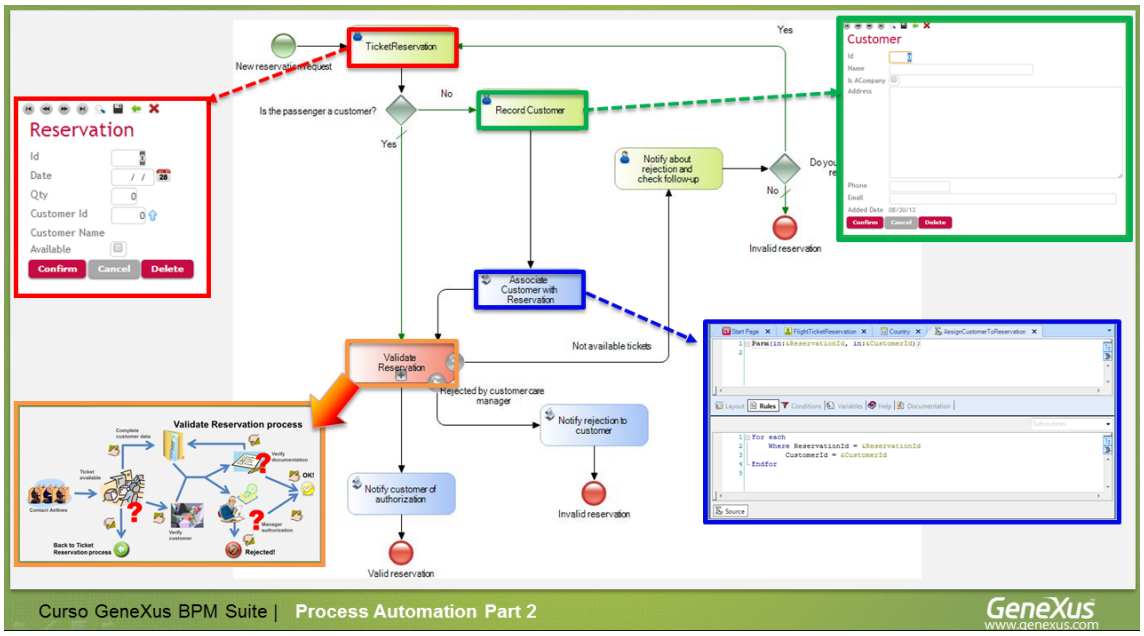
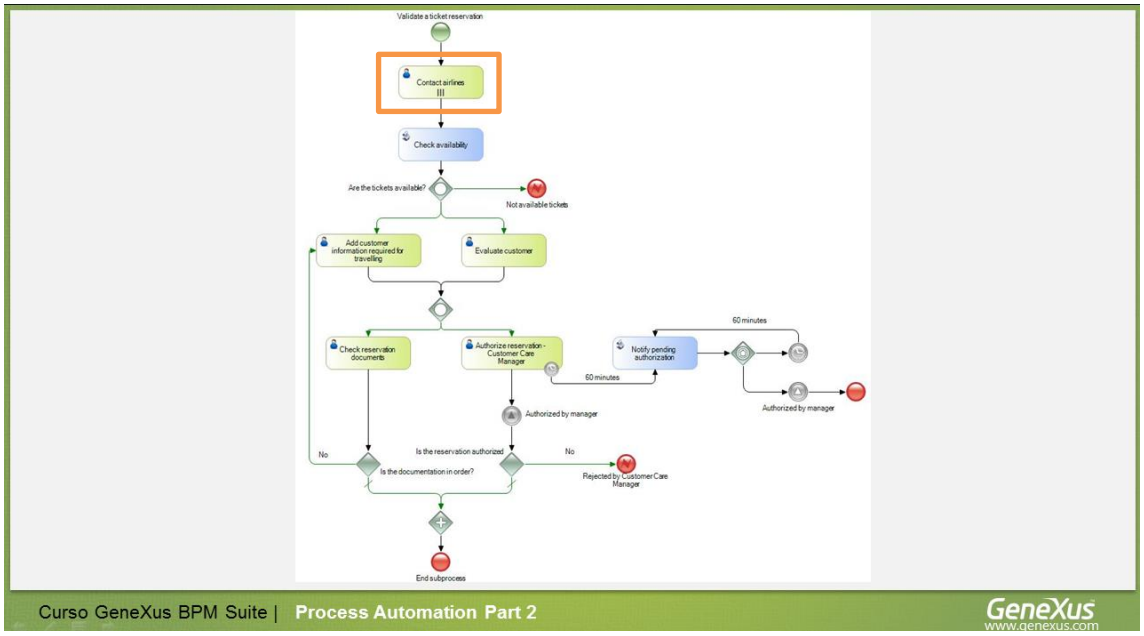
Tareas multi-instanciadas y mapeo de datos relevantes

En videos previos, hemos asociado las tareas del diagrama de reserva de pasajes de la Agencia de Viajes a objetos GeneXus, convirtiendo el modelo del proceso en una aplicación funcional.



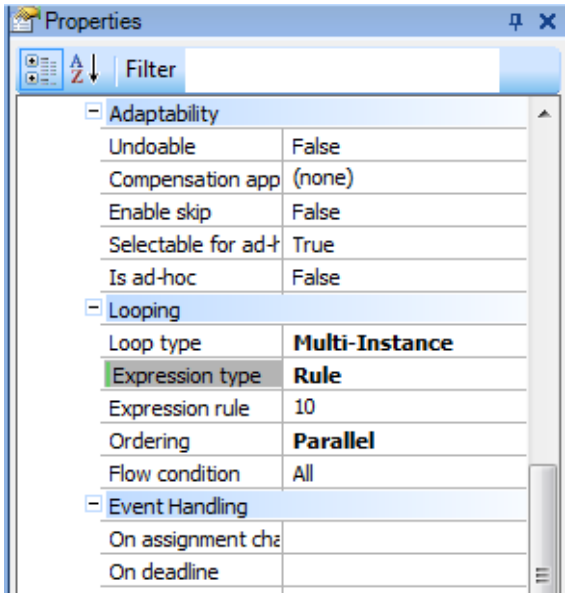
Continuaremos haciendo lo mismo con el diagrama de validación de la reserva, subproceso del proceso de reserva de pasajes.

Si analizamos el diagrama ValidateReservation, vemos que la primer tarea que se ejecutará es la de contactar aerolíneas.



Esta tarea tiene la particularidad de que va a ser ejecutada un cierto número de veces, ya que es necesario contactar a varias aerolíneas y esto incluso podría llegar realizarse simultáneamente, por varios usuarios diferentes.

Si vamos a sus propiedades, vemos que la propiedad **Loop type** tiene el valor: Multi-Instance, la propiedad **Ordering** tiene el valor Parallel, la propiedad **Expression type** tiene el valor Rule y que la propiedad **Expression rule** tiene el valor: 10.

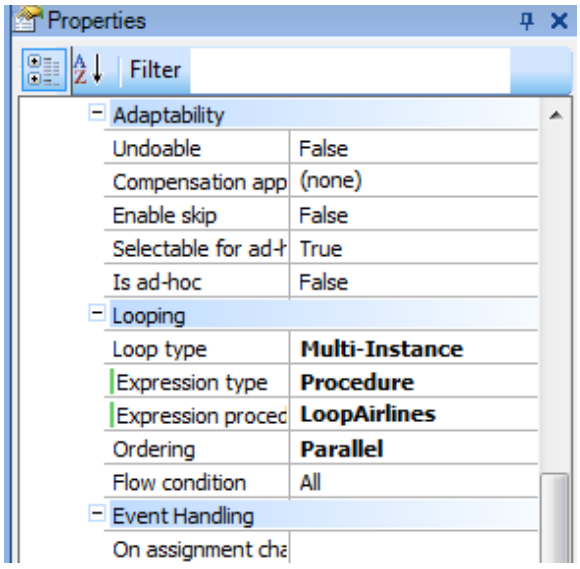


Esto significa que la tarea **se repetirá exactamente 10 veces en paralelo**, que fue la idea manejada en la etapa de modelado. Además, como la propiedad **Flow Condition** tiene el valor All, la tarea ContactAirlines se dará por terminada solamente cuando se terminen de ejecutar las 10 instancias.

Sin embargo, analizando la tarea más en profundidad con el personal de la agencia de viajes, concluimos que la cantidad de veces que se tiene que ejecutar la tarea depende de la cantidad de aerolíneas con las que trabaja la Agencia en ese momento y esta cantidad puede variar con el tiempo.

Para saber cuántas aerolíneas tiene registradas la agencia, podemos utilizar un procedimiento que recorra la tabla de las aerolíneas de la agencia de viajes y nos devuelva la cantidad de aerolíneas registradas.

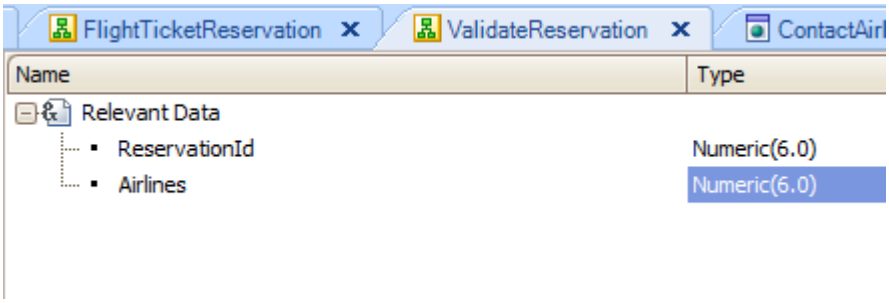
Para implementar esto, cambiamos la propiedad **Expression type** a **Procedure** y en la propiedad **Expression procedure** seleccionamos al procedimiento **LoopAirlines**.



Si abrimos el source del procedimiento, vemos que tiene un For Each que recorre la tabla Airlines y cuenta la cantidad de aerolíneas registradas.

```
1  &ArrayOfAirlines = &WorkflowProcessInstance.GetApplicationDataByName("Airlines")
2  &i=0
3  For each // Airlines
4  Defined by AirlineName
5      &i =&i + 1
6      &ArrayOfAirlines.SetValue(&i, AirlineId.ToString())
7  endfor
8
9  &numberofinstances = &i
10
```

Además, carga los identificadores de las aerolíneas en un array, que fue definido como dato relevante del diagrama ValidateReservation.



La forma que se accede a este dato relevante desde el procedimiento es utilizando métodos de la API del motor de Workflow. Veremos esto en detalle más adelante, en otro video.

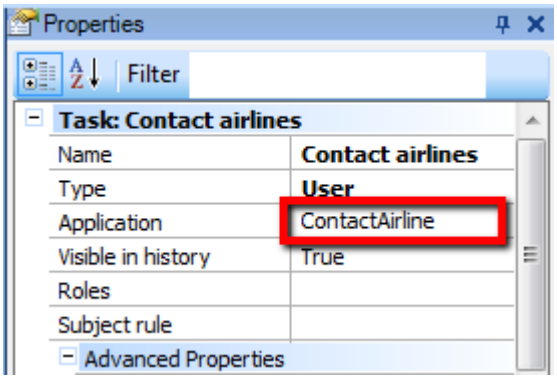
```
1  &ArrayOfAirlines = &WorkflowProcessInstance.GetApplicationDataByName("Airlines")
2  &i=0
3  For each // Airlines
4  Defined by AirlineName
5      &i =&i + 1
6      &ArrayOfAirlines.SetValue(&i, AirlineId.ToString())
7  endfor
8
9  &numberofinstances = &i
10
```

Como mencionamos antes, la cantidad de aerolíneas determina la cantidad de instancias que se crearán de la tarea ContactAirlines, por lo que este valor es devuelto por el procedimiento a la tarea ContactAirlines, en el último parámetro de la regla Parm.

```
1  parm(in:&WorkflowProcessDefinition,in:&WorkflowProcessInstance,in:&WorkflowWorkitem,out:&numberofinstances);
```

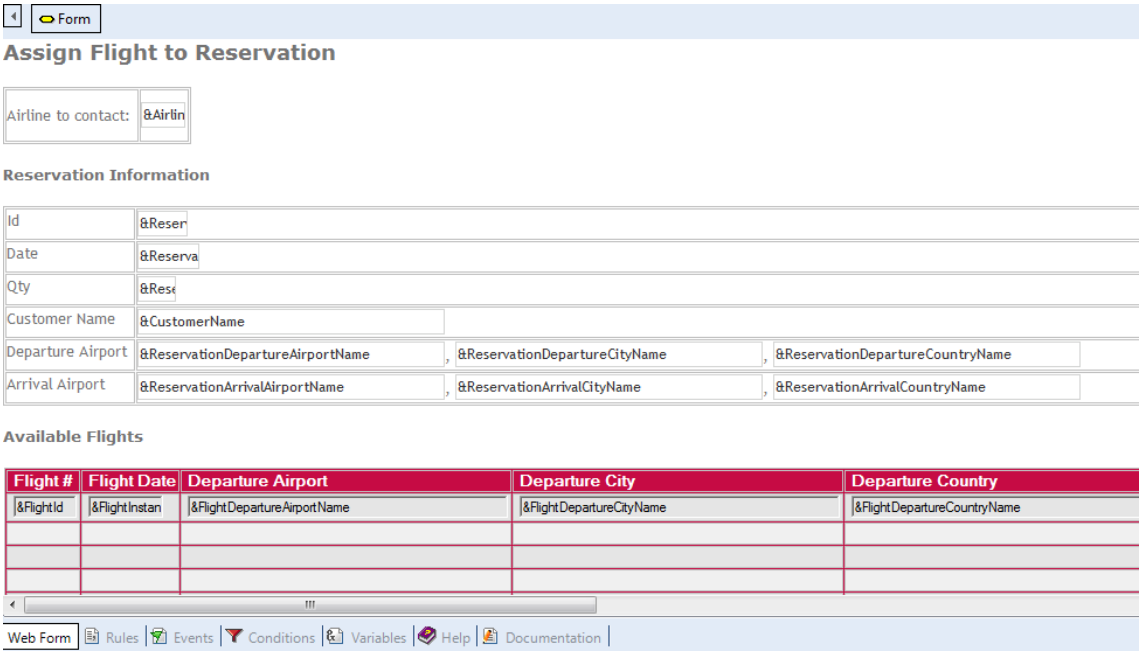
Resumiendo, para definir una tarea con múltiples instancias, asignamos la propiedad LoopType en el valor Multi-Instance y para definir la cantidad de veces que dicha tarea se instancia, usamos la propiedad Expression type en Rule y asignamos la cantidad en la propiedad Expression Rule o bien asignamos la propiedad Expression Type en Procedure y utilizamos un procedimiento que devuelve la cantidad de veces que se instanciará la tarea, como vimos en este último caso.

Volviendo al diagrama, la tarea ContactAirlines tendrá asociado un objeto GeneXus del tipo webpanel, que se ejecutará cada vez que se ejecuta la tarea. Su nombre es ContactAirline. Este webpanel permitirá elegir por cada aerolínea, el vuelo adecuado para la reserva.



Al iniciarse la ejecución del webpanel, se asociará internamente la instancia de la tarea en ejecución a una de las aerolíneas, de forma que cada vez que se inicie una nueva instancia de la tarea, se contactará a una aerolínea diferente, de las que están registradas por la agencia.

El webpanel muestra los datos de la reserva y los vuelos que la aerolínea seleccionada tiene disponibles, para la fecha de la reserva.

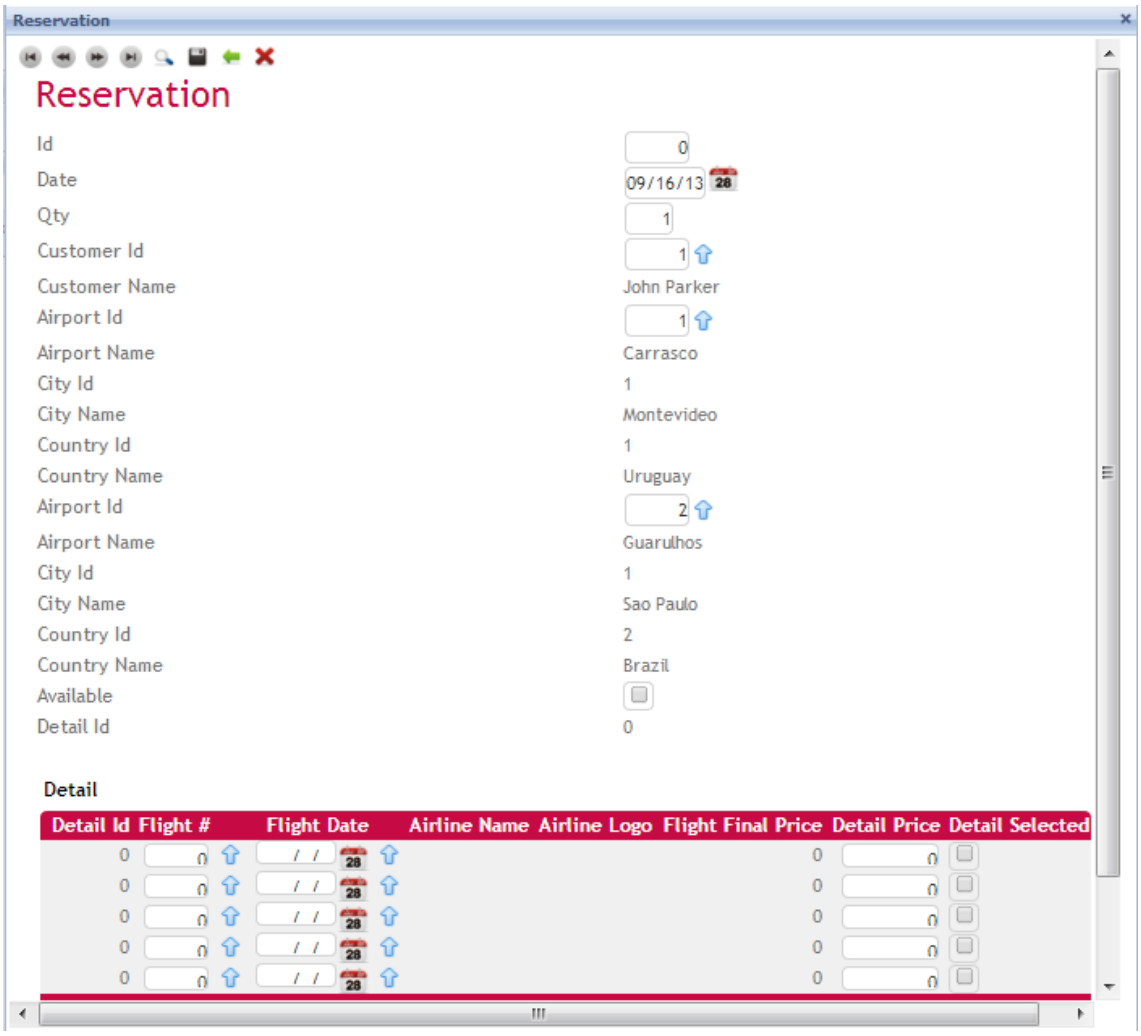


El operario de la agencia podrá seleccionar el vuelo que desee asociar a la reserva.

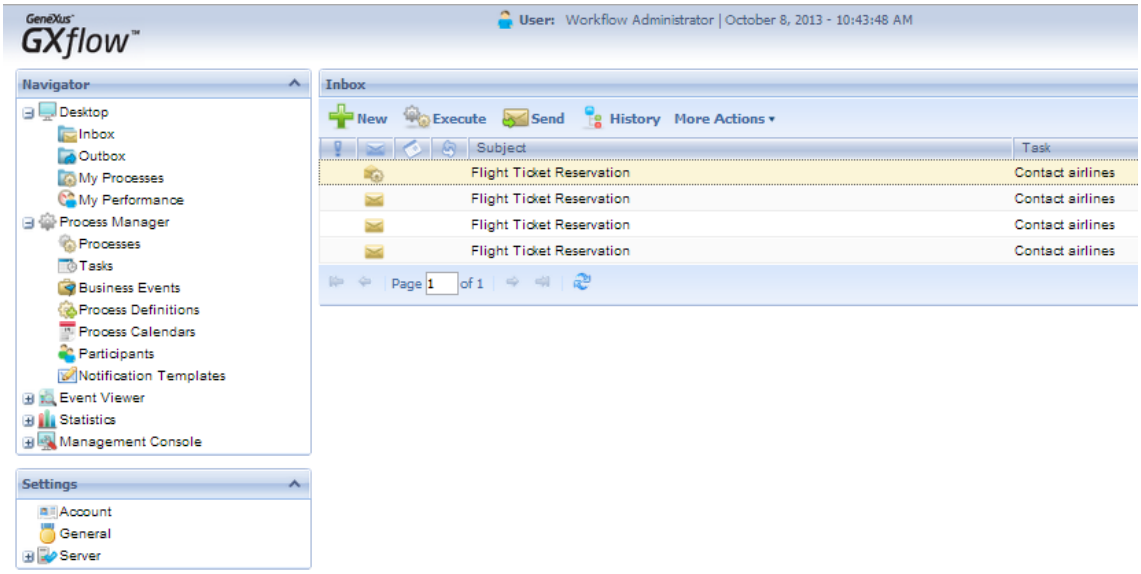
Vamos a ver esto en ejecución.

Sobre la solapa del diagrama FlightTicketReservation, damos botón derecho y elegimos Run.

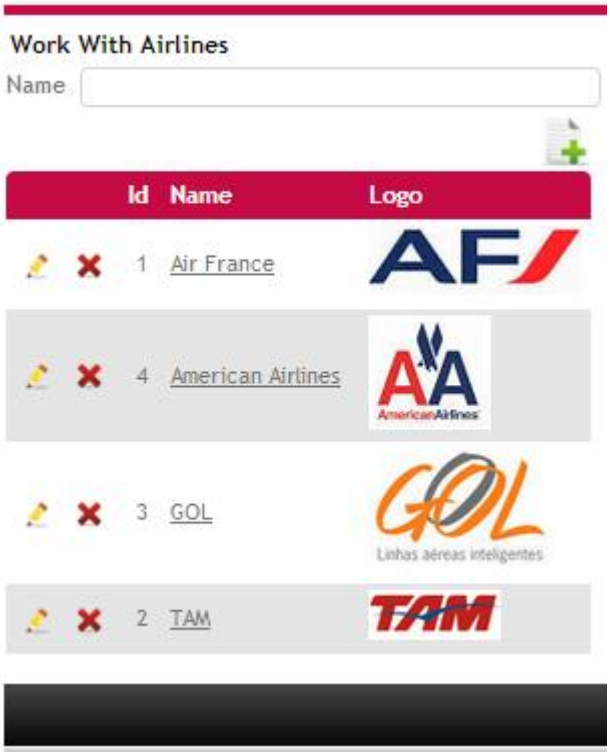
Ejecutamos la tarea TicketReservation e ingresamos una reserva para el día de hoy, para el cliente 1, John Parker, que desea ir desde el aeropuerto de Carrasco, en Montevideo, hasta el aeropuerto de Guarulhos, en San Pablo. Presionamos Confirmar y cerramos la pantalla.



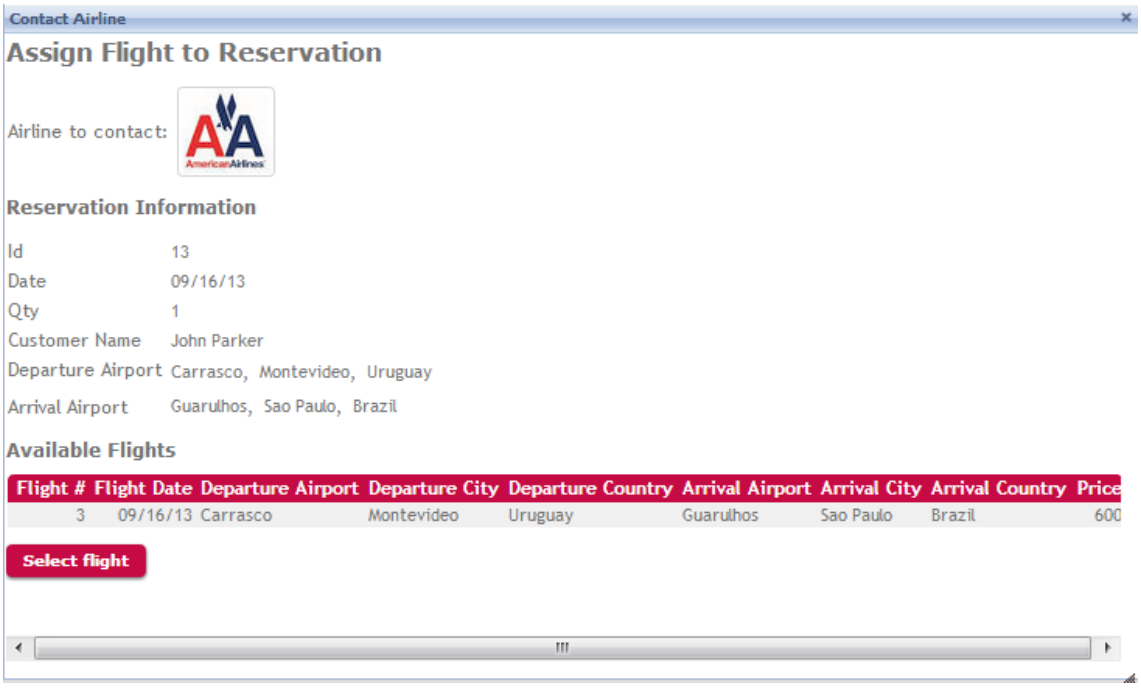
Presionamos Send para enviar la tarea y vemos que aparecen cuatro tareas ContactAirlines pendientes.



Esto es porque tenemos 4 aerolíneas para contactar y se creó una instancia de la tarea ContactArlines para cada una de las aerolíneas registradas en la Agencia.

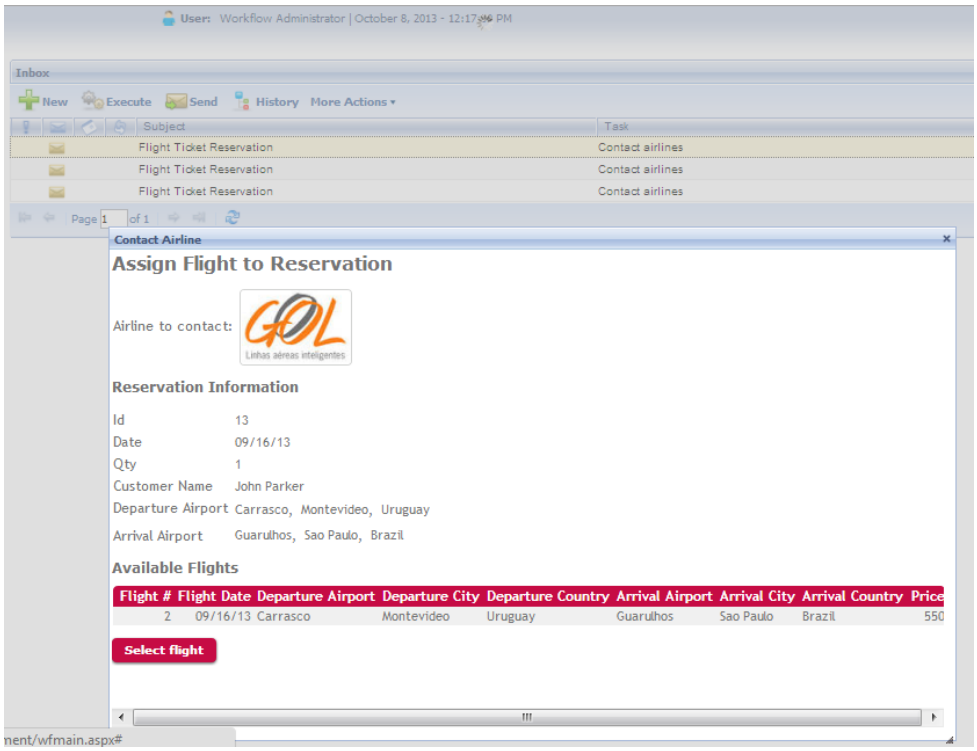


Si hacemos doble clic en la primera tarea pendiente, se abre una pantalla para contactar a la primera aerolínea. Vemos que tiene un vuelo disponible para el día, origen y destino requeridos por la reserva, así que seleccionamos el vuelo y presionamos Select Flight.



De esta forma asignamos un posible vuelo, que cumple con la reserva solicitada.

Cerramos la ventana y completamos la tarea, por lo cual desaparece del inbox como tarea pendiente. Si ejecutamos la siguiente tarea, vemos que se asigna una aerolínea diferente y será así para cada instancia de la tarea ContactAirlines.



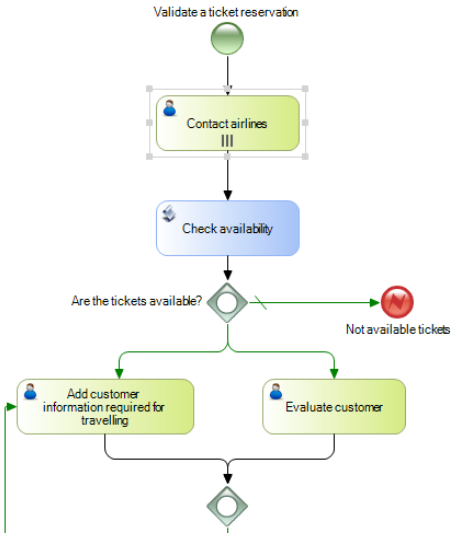
Eso se resuelve en el objeto webpanel, ya que a partir del dato relevante Airlines (del tipo array) que guarda los identificadores de aerolíneas, cada vez que se inicia el webpanel, se obtiene un elemento del array diferente para cada instancia de la tarea ContactAirlines.

```
Event Start
    &ContactedAirlinesWorkflowApplicationData = &WorkflowContext.ProcessInstance.GetApplicationDataByName("ContactedAirlines")
    &AirlinesWorkflowApplicationData = &WorkflowContext.ProcessInstance.GetApplicationDataByName("Airlines")

    // Get the corresponding Airline from the workitem index
    &i = &WorkflowContext.Workitem.Index
    &AirlineId = &AirlinesWorkflowApplicationData.GetValue(&i).ToNumeric()
    For each
        Where AirlineId=&AirlineId
        &AirlineName = AirlineName.Trim()
        &AirlineLogo = AirlineLogo
    Endfor
Endevent

```

Continuando con el proceso de validación de la reserva, luego de contactar las aerolíneas, se verifica mediante la tarea **CheckAvailability** que se haya podido encontrar al menos un vuelo que satisfaga la reserva.



Si abrimos el procedimiento **CheckReservationFlights** y vamos al source, vemos que tiene un For Each que recorre la tabla de detalle de la reserva y verifica que haya al menos un vuelo seleccionado para la reserva. En caso afirmativo, asigna el valor True a la variable &ReservationAvailable.

```
1  //Check if the reservation has at list one assigned flight
2  &ReservationAvailable = False
3  For each // RESERVATIONDETAIL
4       Defined by ReservationDetailSelected
5       &ReservationAvailable = True
6       Exit
7  Endfor
8
```

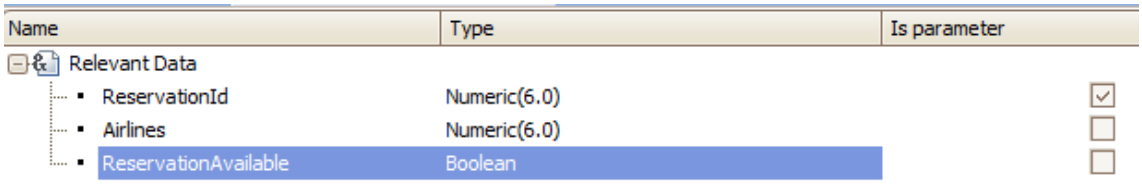
Esta variable se retorna como último parámetro de la regla Parm del procedimiento.

```
1  Parm(in:&ReservationId, out:&ReservationAvailable);
2
```

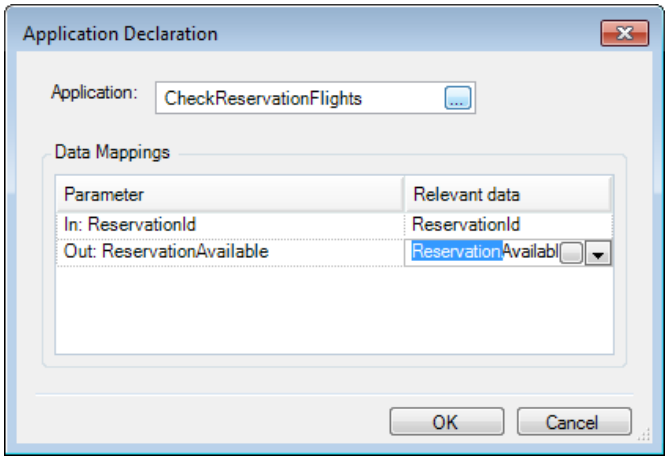
Si a esta variable la llamamos exactamente igual que un dato relevante, el motor de workflow cargará automáticamente el dato relevante con el valor de la variable.

En los objetos procedimiento, el mapeo de valores entre los datos relevantes y las variables presentes en la regla Parm, es válido tanto para variables de entrada como de salida, mientras que en el caso de objetos webpanels, el mapeo de los valores es solamente válido para variables de entrada.

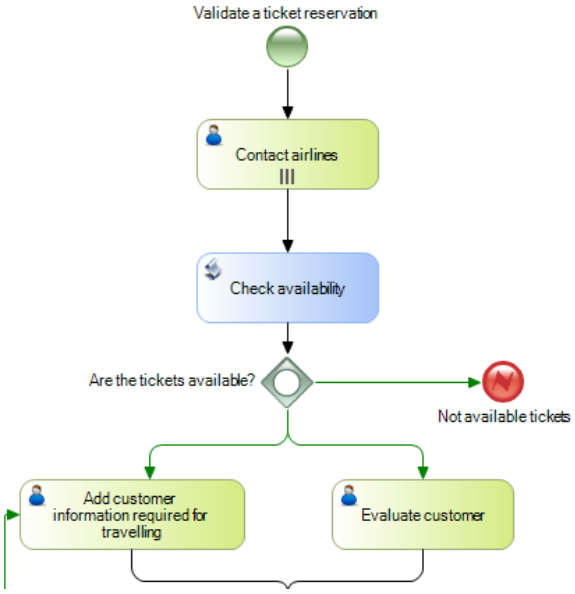
Así que vamos al diagrama ValidateReservation, seleccionamos la solapa RelevantData y creamos el dato relevante **&ReservationAvailable** del tipo boolean y desmarcamos el check box "IsParameter" porque ese dato no es un parámetro del objeto diagrama.



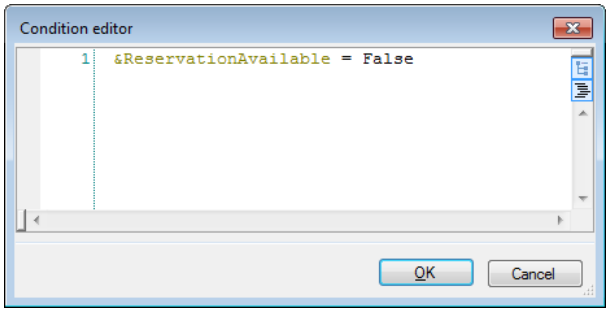
Por último, asociamos el procedimiento **CheckReservationFlights** a la tarea batch **CheckAvailability** y mapeamos los datos relevantes ReservationId y ReservationAvailable.



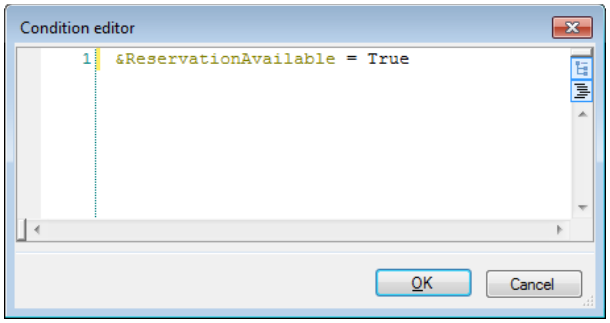
Volviendo al diagrama, una vez que el procedimiento establece si la reserva está disponible o no, el inclusive gateway "Are the tickets available?" debería chequear el valor del dato relevante que cargamos.



Para hacerlo, hacemos doble clic en el conector que sale a la derecha del inclusive Gateway y escribimos &ReservationAvailable=False. En la propiedad **Text** escribimos "Tickets not available".

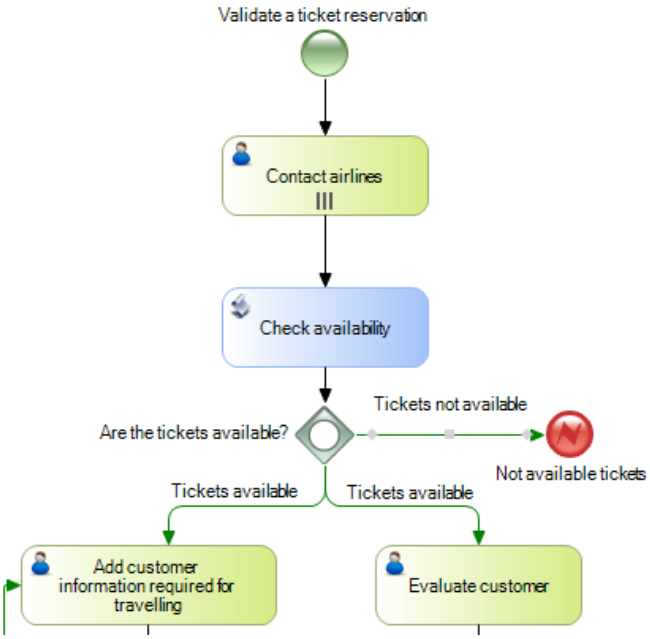


Lo mismo hacemos con los dos conectores que salen hacia abajo del inclusive Gateway, asignándoles la condición &ReservationAvailable=True y en la propiedad **Text**: "Tickets available".



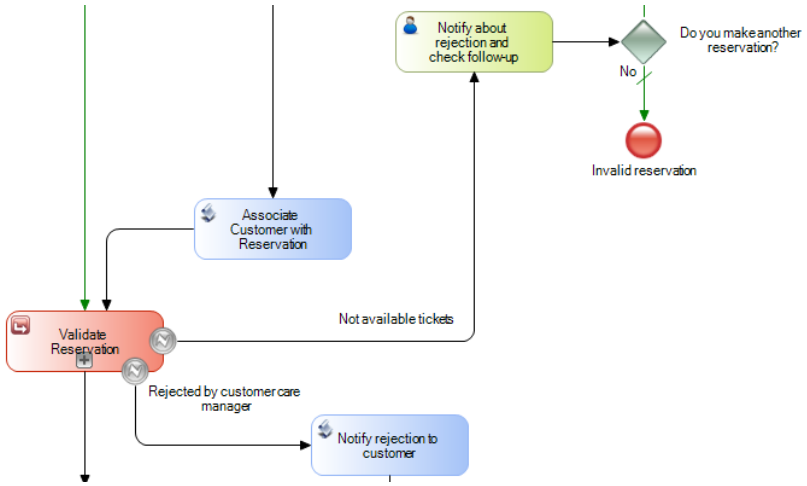
En las expresiones de condición de un Gateway podemos incluir datos relevantes, constantes (como el valor True de este caso), valores de dominios enumerados y atributos de la tabla extendida de las transacciones asociadas al diagrama.

Con las definiciones que hicimos, si hay vuelos disponibles para la reserva, el flujo seguirá hacia abajo del Gateway y si no hay vuelos para la reserva, se seguirá hacia la derecha, terminando en el **Error End Event** llamado "Not available tickets".



Este tipo de evento de finalización con error, nos permite finalizar el subproceso de validación de la reserva y enviar la comunicación del error al proceso principal de reserva de pasajes.

Si observamos el proceso principal, vemos que también hay un símbolo de evento intermedio de error, con la misma etiqueta "Not available tickets", que está conectado a una tarea interactiva donde se le notifica de la situación al cliente.



El evento intermedio de error es del tipo "catch", mientras que el evento de fin de error del subproceso, es del tipo "throw".

Esta es la forma en que podemos saber desde el proceso principal, cuál fue exactamente la causa de la finalización del subproceso y actuar en consecuencia.

En el próximo video continuaremos con el subproceso de validación de la reserva, con las tareas interactivas "Add customer information required for traveling" y "Evaluate Customer"que se ejecutarán simultáneamente.