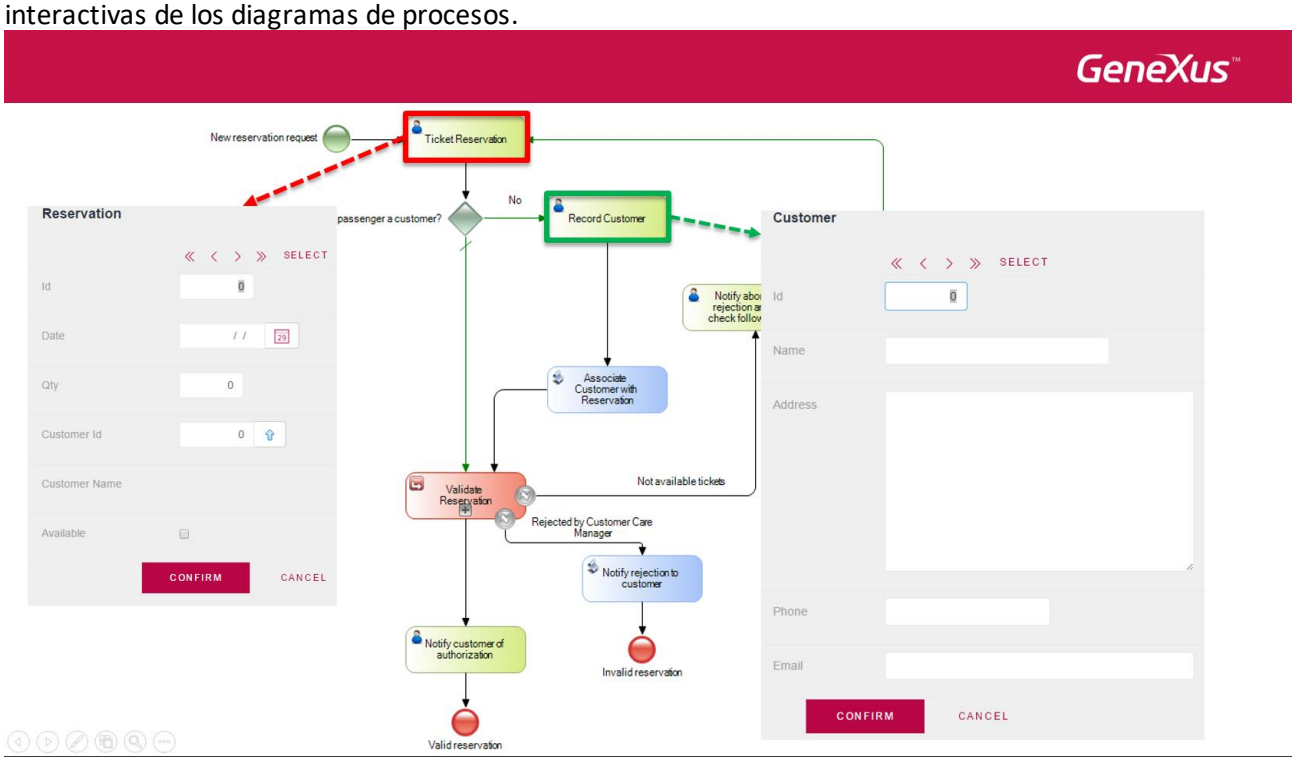
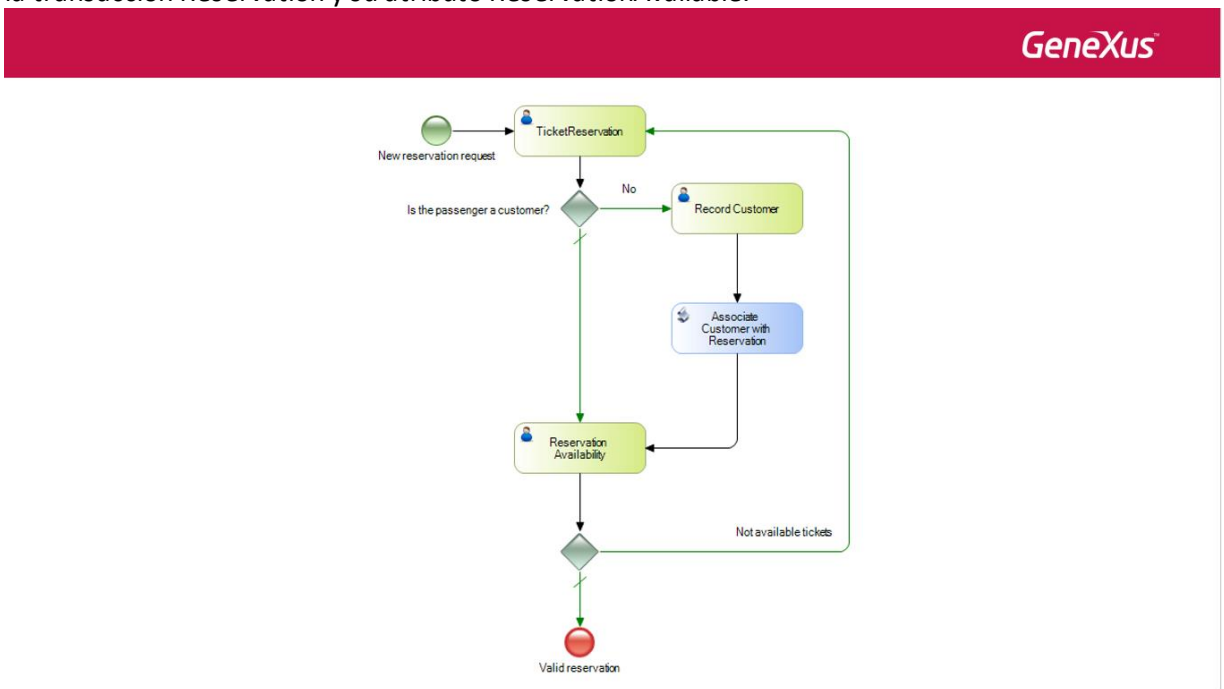
Ejecutando un proceso de negocios en un dispositivo móvil

Hasta ahora hemos asociado objetos GeneXus que se ejecutaban en una página web, a las tareas interactivas de los diagramas de procesos.



Pero también es posible asociar a las tareas de usuario, objetos GeneXus que se ejecuten en un Smart Device. Por ejemplo, supongamos que el proceso de ingresar una reserva de la agencia de viajes, lo queremos ejecutar desde un dispositivo móvil.

Vamos a simplificar el proceso de reserva de pasajes que habíamos diseñado. Quitamos el subproceso de validación y algunos símbolos que no usaremos. Para especificar la disponibilidad de la reserva utilizaremos la transacción Reservation y su atributo ReservationAvailable.

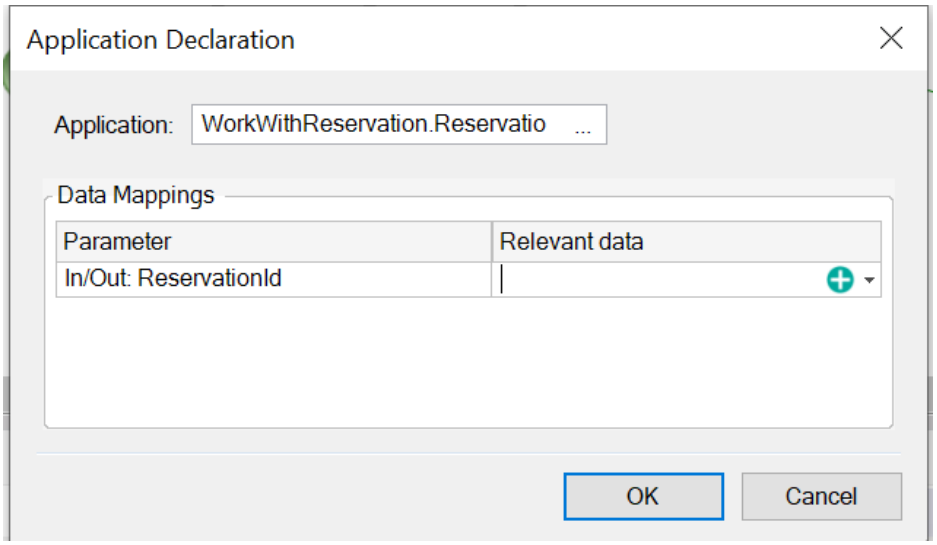


Por último seleccionamos la tarea TicketReservation y para este ejemplo, en la propiedad Roles, borramos el valor y lo dejamos vacío. Lo mismo hacemos con el None Start Event.

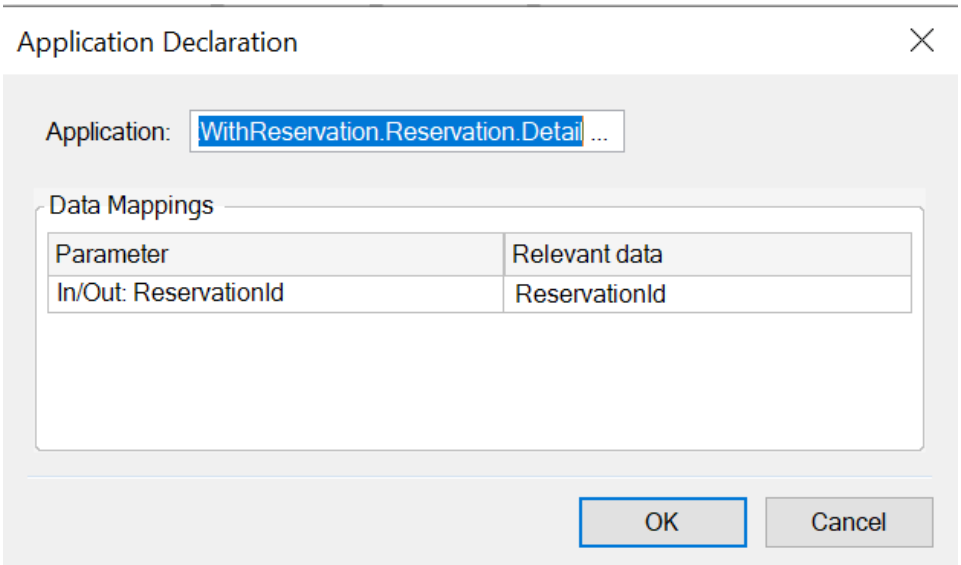
Lo primero que vamos a hacer ahora es aplicar el patrón Work With a las transacciones Reservation y Customer, que son las involucradas en el proceso. Para eso, seleccionamos ambas transacciones, damos botón derecho, elegimos Apply Pattern y Work With.

Vemos que bajo el nodo de la transacción Customer aparece el objeto SD: WorkWithCustomer y bajo la transacción Reservation, el objeto SD: WorkWithReservation.

Ahora asociaremos a la tarea TicketReservation, la aplicación SD que generó el patrón. Para eso vamos a las propiedades de la tarea y en la propiedad Object presionamos el botón y elegimos WorkWithReservation.



Presionamos tabulador y asociamos el dato relevante ReservationId.

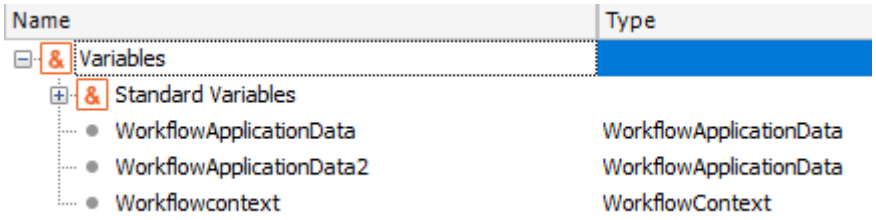


Presionamos OK.
Algo muy importante a tomar en cuenta cuando asociamos objetos Smart Devices, es que **los datos relevantes no se mapean automáticamente entre el diagrama y el objeto SD**. Para hacer esta asociación deberemos contar con un procedimiento.

Abrimos el objeto procedimiento *ReservationMapRelevantData* que creamos previamente.
En la sección reglas vemos que definimos una regla Parm que recibe a los identificadores de reserva y cliente, en los atributos ReservationId y CustomerId respectivamente.

```
1  Parm(in:ReservationId, in:CustomerId);
```

También definimos las siguientes variables:



Y en el source, utilizando la API de Workflow, obtenemos el dato relevante ReservationId por su nombre y lo asociamos al atributo ReservationId que recibimos por parámetro.
Lo mismo hacemos con el dato relevante CustomerId y el atributo CustomerId.

```
1  &WorkflowApplicationData = &Workflowcontext.ProcessInstance.GetApplicationDataByName("ReservationId")
2  &WorkflowApplicationData.NumericValue = ReservationId
3
4  &WorkflowApplicationData2 = &Workflowcontext.ProcessInstance.GetApplicationDataByName("CustomerId")
5  if not CustomerId.IsNull()
6      &WorkflowApplicationData2.NumericValue = CustomerId
7  EndIf
8
9  Commit
```

No debemos olvidarnos de incluir el Commit, cuando trabajamos con tipos de datos Workflow.

Como preguntamos por el valor isNull() de CustomerId, vemos que agregamos la siguiente regla a la transacción Reservation:

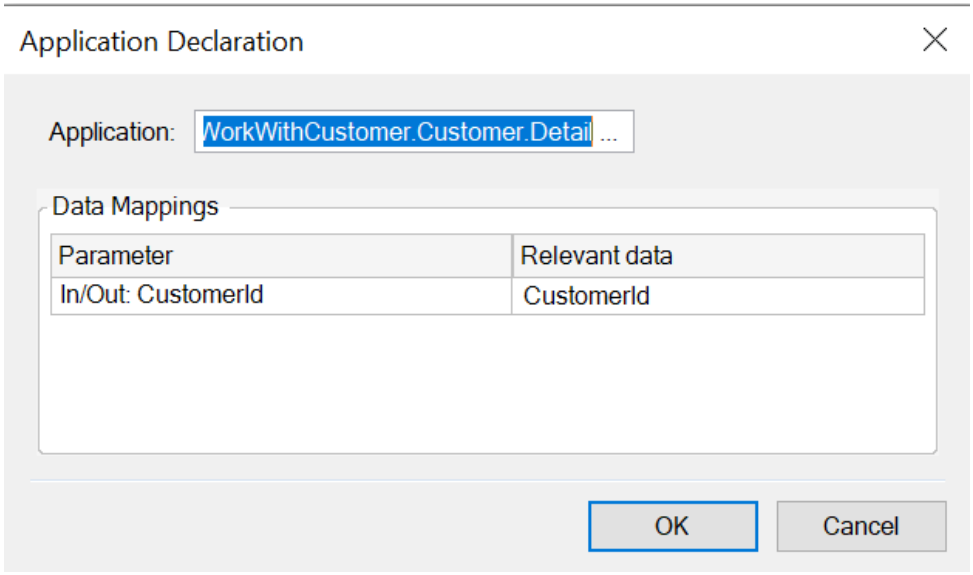
```
6  CustomerId.SetNull() If CustomerId.IsEmpty();
```

Ahora abrimos el objeto WorkWithReservation, vamos a su sección Detail / General y modificamos el evento Save, agregando la invocación al procedimiento que acabamos de crear.

```
Event 'Save'
    Composite
        SDActions.Save()
        ReservationMapRelevantData.Call(ReservationId,CustomerId)
        return
    EndComposite
EndEvent
```

De esta manera cuando insertamos una reserva nueva desde la aplicación SD, se mapearán los datos relevantes correspondientes.

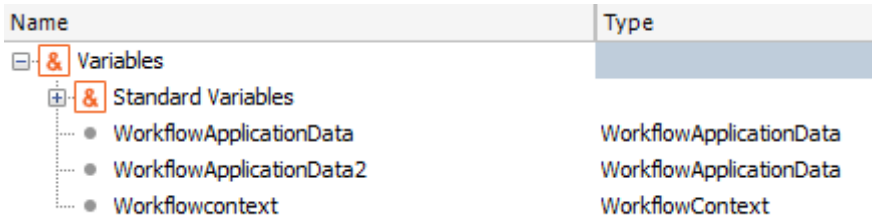
A continuación, asociamos la tarea RecordCustomer a la aplicación WorkWithDevicesCustomer, y asignamos el dato relevante CustomerId:



En forma similar a lo sucedido con las reservas, tenemos que contar con un procedimiento para asociar el dato relevante CustomerId al parámetro CustomerId de la transacción Customer.
Para esto creamos previamente el procedimiento de nombre *CustomerMapRelevantData*. En la sección de reglas vemos a la regla Parm que recibe el identificador del cliente en el atributo CustomerId:

```
1 Parm(in:CustomerId);
```

Vemos las variables de tipos de datos Workflow definidas:



Y el source implementado donde obtenemos el dato relevante CustomerId por su nombre y lo asociamos al atributo CustomerId que recibimos por parámetro

```
1 &WorkflowApplicationData2 = &Workflowcontext.ProcessInstance.GetApplicationDataByName("CustomerId")
2 &WorkflowApplicationData2.NumericValue = CustomerId
3
4 Commit
```

Ahora vamos al objeto WorkWithsCustomer, en la sección Detail / General y modificamos el evento Save, agregando la invocación al procedimiento CustomerMapRelevantData.

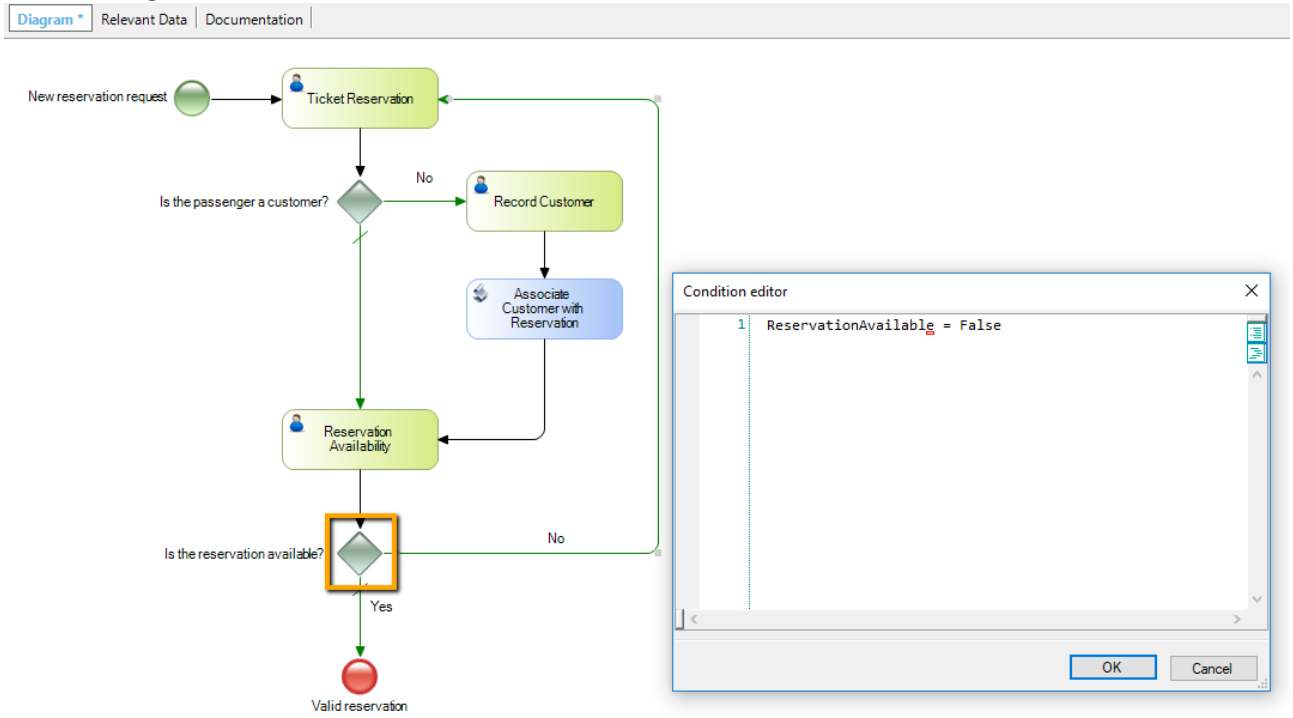
```
Event 'Save'
    Composite
        SDActions.Save()
        CustomerMapRelevantData.Call(CustomerId)
        return
    EndComposite
EndEvent
```


Continuando con nuestro diagrama, vamos a asignar la tarea ReservationAvailability al objeto

WorkWithReservations, en forma análoga a como hicimos antes con la tarea Reservation.

Si la reserva está disponible termina el proceso, pero si no lo está debemos ingresar una reserva nueva. Esta evaluación la hacemos con el exclusive Gateway.

Si hacemos doble clic sobre el conector que une el Gateway con la tarea TicketReservation, vemos que ya teníamos ingresada la condición necesaria.



Para poder ejecutar el diagrama de procesos que construimos, debemos importar y configurar el **cliente GXflow para Smart Devices**. Aquí ya lo hemos hecho, pero los detalles los encontrará en el siguiente link en pantalla:

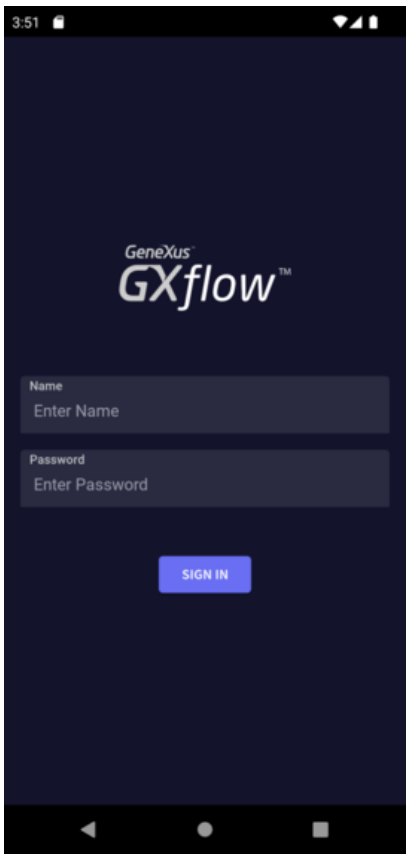
HowTo: Configuring GXflow Client For Smart Devices

Una vez importado y configurado el cliente GXflow para SD, es necesario que tengamos una invocación a cada objeto SD usado en nuestro diagrama de procesos.
Para eso, debemos agregar el siguiente código al dashboard WorkflowSDClient:

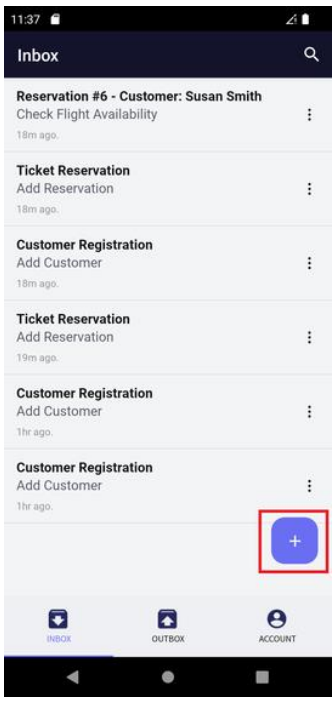
```
Event 'DummyCalls'
        WorkWithDevicesReservation.Reservation.Detail(1)
        WorkWithDevicesReservation.Reservation.List()
        WorkWithDevicesCustomer.Customer.Detail(1)
        WorkWithDevicesCustomer.Customer.List()
EndEvent
```

Ahora estamos listos para ejecutar nuestro proceso de reserva de pasajes en una plataforma mobile.
Primero hacemos un Build All…
Y ahora presionamos F5.

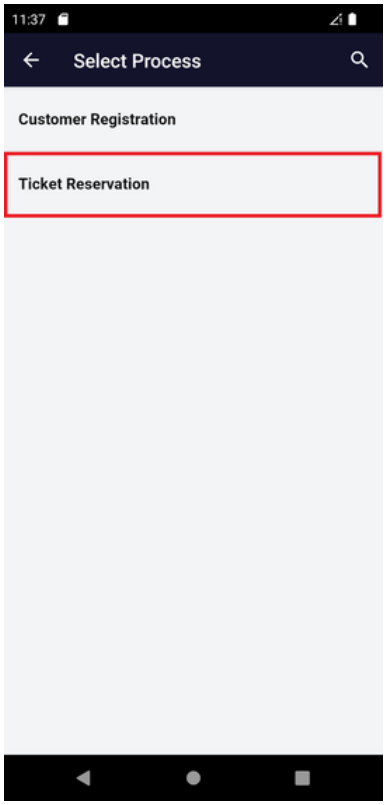
Vemos que el emulador de Android se ejecutó automáticamente mostrando la pantalla de login:



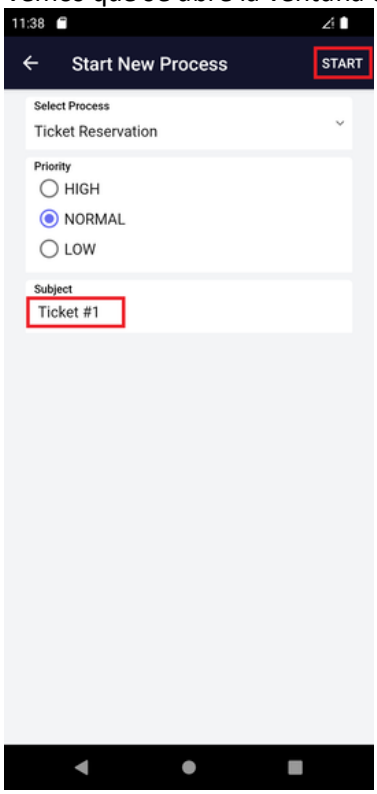
Vamos a loguearnos con el usuario administrador de workflow, así que ingresamos como usuario: WFADMINISTRATOR y lo mismo como password. Al ingresar vemos una pantalla que nos muestra la bandeja de entrada y presionamos el botón de “+” para instanciar un proceso.



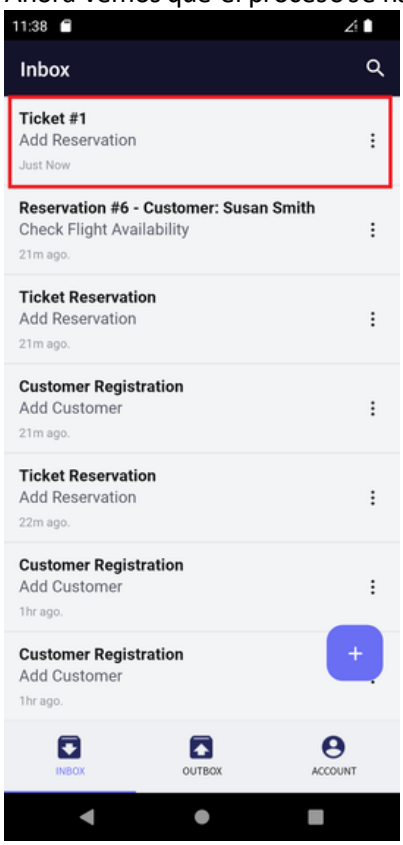
Ahora seleccionamos el proceso FlightTicketReservation



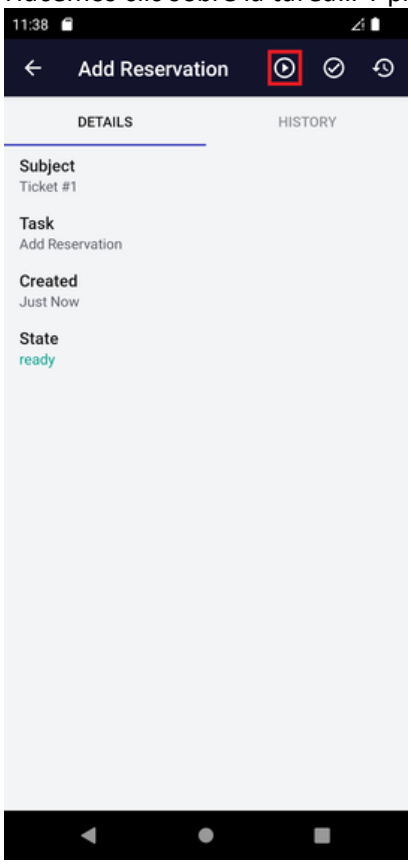
Vemos que se abre la ventana del proceso. Presionamos el botón de Start para iniciar el mismo.



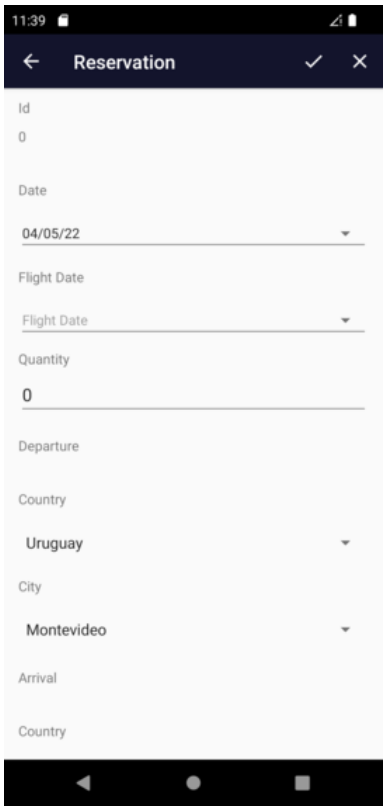
Ahora vemos que el proceso se ha iniciado y que tenemos la tarea TicketReservation pendiente de ejecutar.



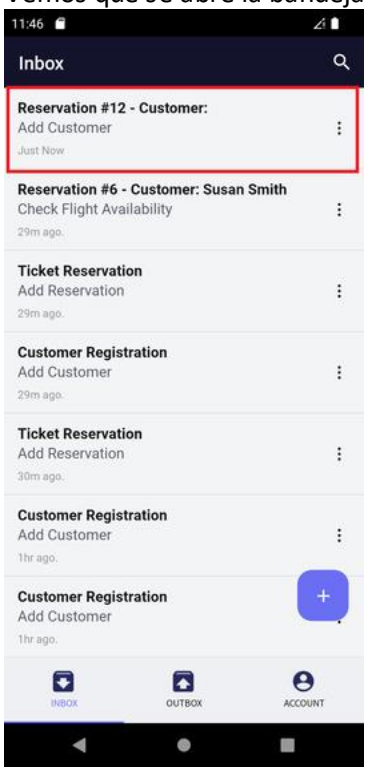
Hacemos clic sobre la tarea… Y presionamos el botón de la flecha para iniciarla.



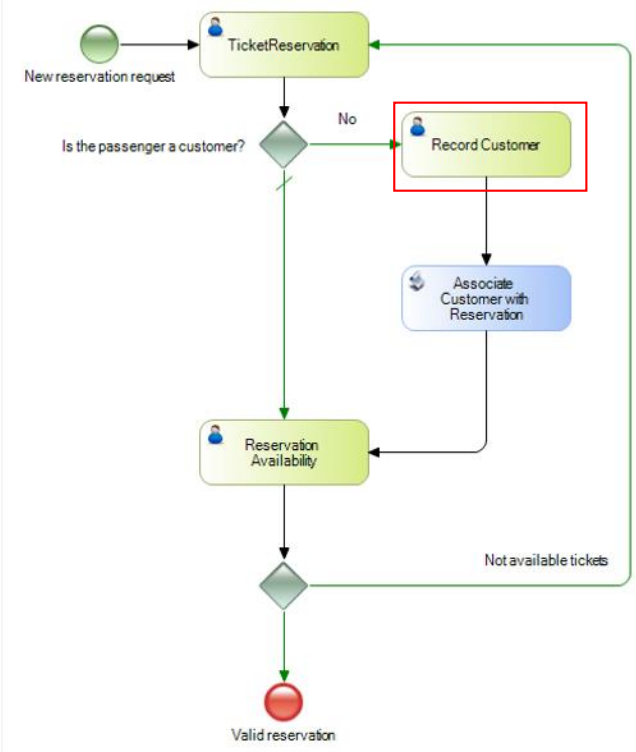
Vemos que se abre el objeto SD para trabajar con reservas, para que ingresemos una reserva. Vamos a ingresar una reserva nueva, dejando el Id sin especificar ya que es autonumerado y dejamos el identificador del cliente vacío.



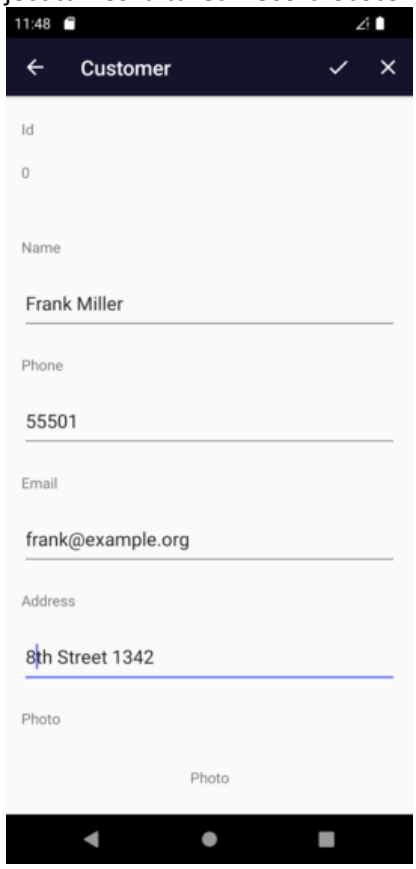
Ahora presionamos el botón de Confirmar. Para finalizar la tarea, elegimos Completar.
Vemos que se abre la bandeja de entrada y ahora la tarea pendiente de ejecución es RecordCustomer.



Como dejamos el cliente sin ingresar, el motor de workflow evaluó la condición del exclusive Gateway “Is the passenger a customer?” y determinó que la siguiente tarea será RecordCustomer, la cual invocará al objeto SD Trabajar con Clientes, para que ingresemos al cliente.

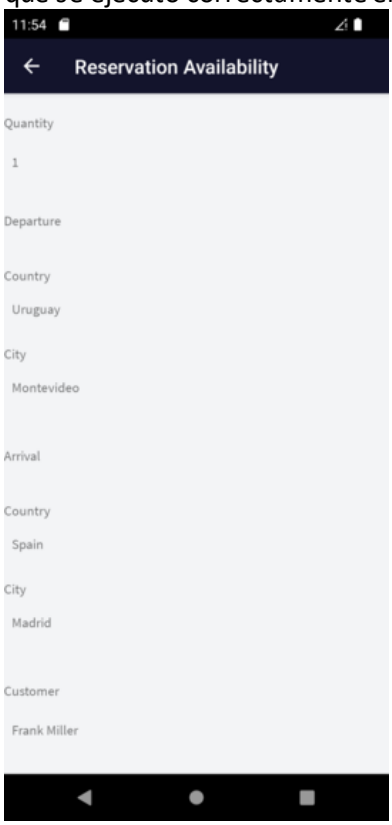


Ejecutamos la tarea Record Customer, ingresamos los datos del cliente y presionamos Confirmar.



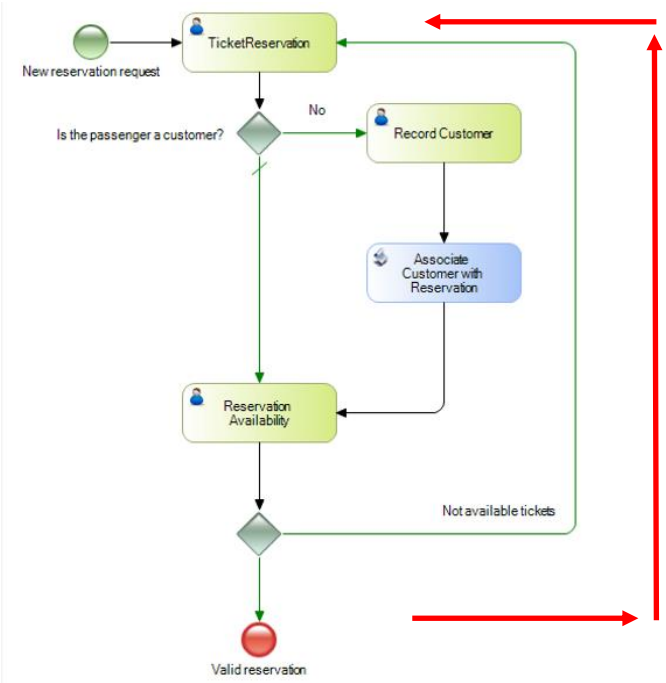
Para finalizar, completamos la tarea Record Customer. Ahora la próxima tarea que aparece como pendiente es ReservationAvailability. Damos click sobre la misma para abrirla.

Y a continuación la ejecutamos. Observemos que el cliente fue exitosamente asignado a la reserva, dado que se ejecutó correctamente el procedimiento asociado a la tarea batch AssociateCustomerToReservation.

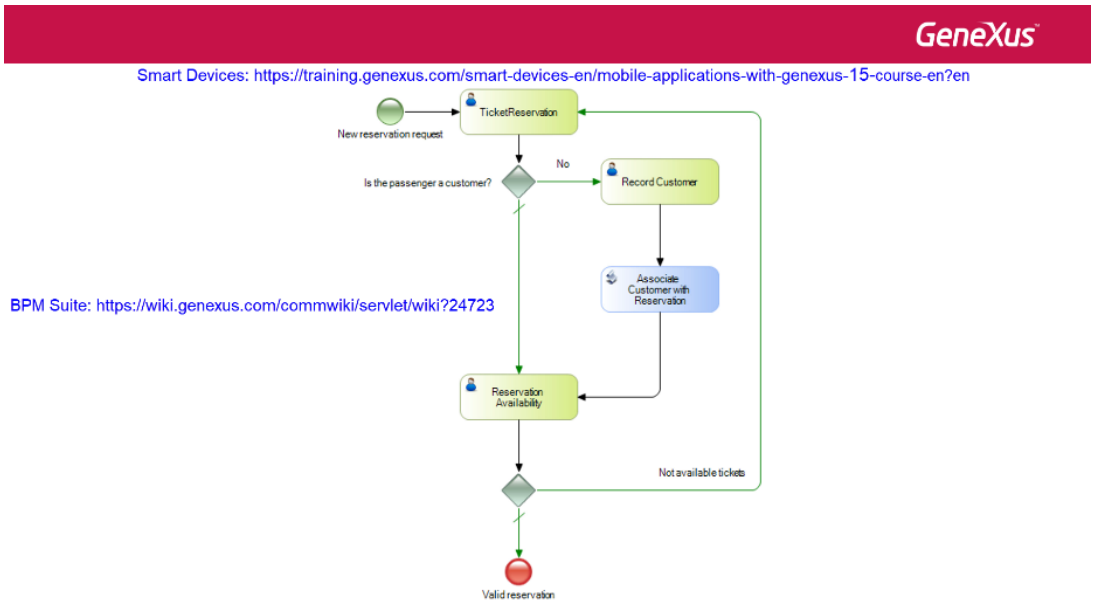


Vamos a marcar que la reserva está disponible y presionamos Confirmar. Para finalizar, completamos la tarea ReservationAvailability.

Ahora la bandeja de entrada no muestra más tareas pendientes, lo que implica que finalizó la ejecución del proceso de reserva de la agencia de viajes. Si hubiéramos marcado que la reserva no estaba disponible, se hubiera ejecutado nuevamente el objeto SD WorkWithReservation, para que ingresáramos una reserva nueva.



Hemos visto así cómo ejecutar un diagrama de proceso de negocios, en un dispositivo móvil.
En particular, hemos asociado a las tareas, los objetos generados por el patrón Work With for Smart Devices, pero también podríamos haber asociado un objeto SD creado por nosotros, como por ejemplo un panel para SD.
En este caso hemos generado la aplicación para Smart Devices en Android y ejecutamos la misma utilizando un emulador, pero es posible prototipar sobre un dispositivo físico y generar aplicaciones para otras plataformas como por ejemplo dispositivos iOS como Ipad o Iphone.
Para saber más sobre aplicaciones Smart Devices, visite el link que se muestra en pantalla:



Y para conocer más posibilidades de la suite BPM de GeneXus, visite el siguiente link del wiki.