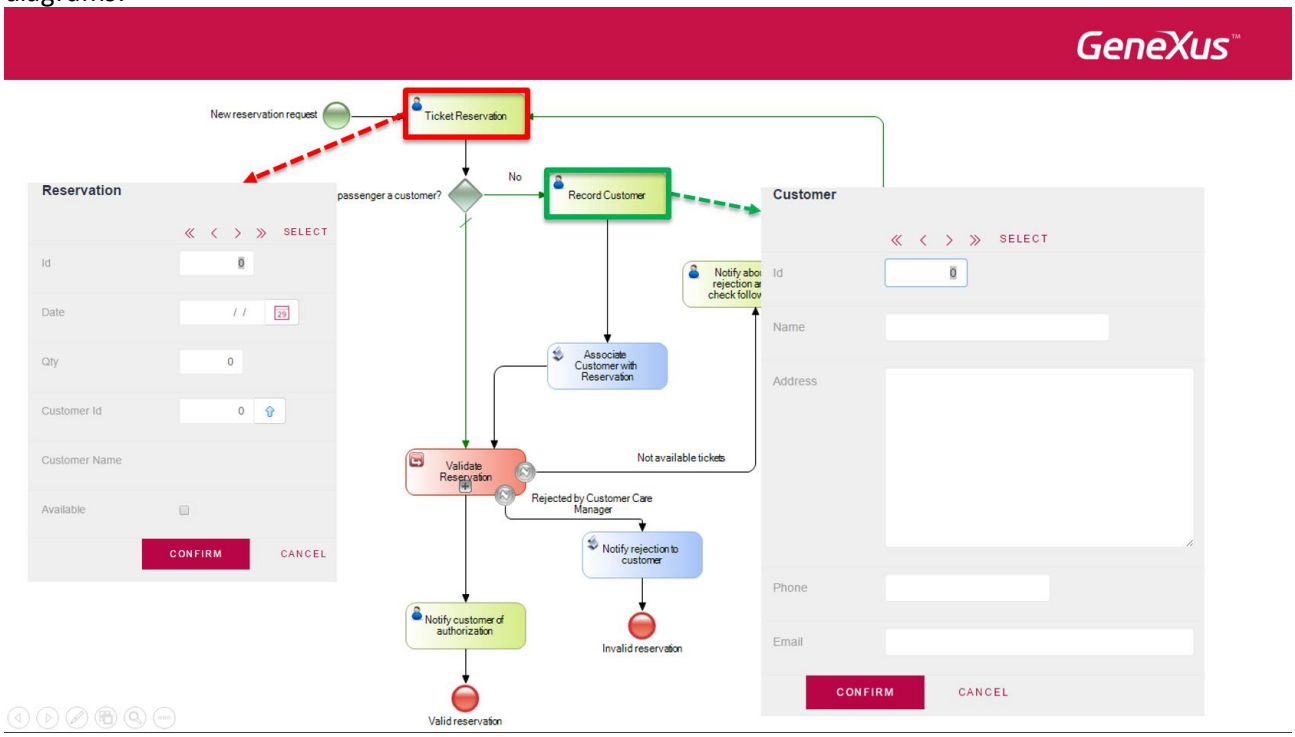
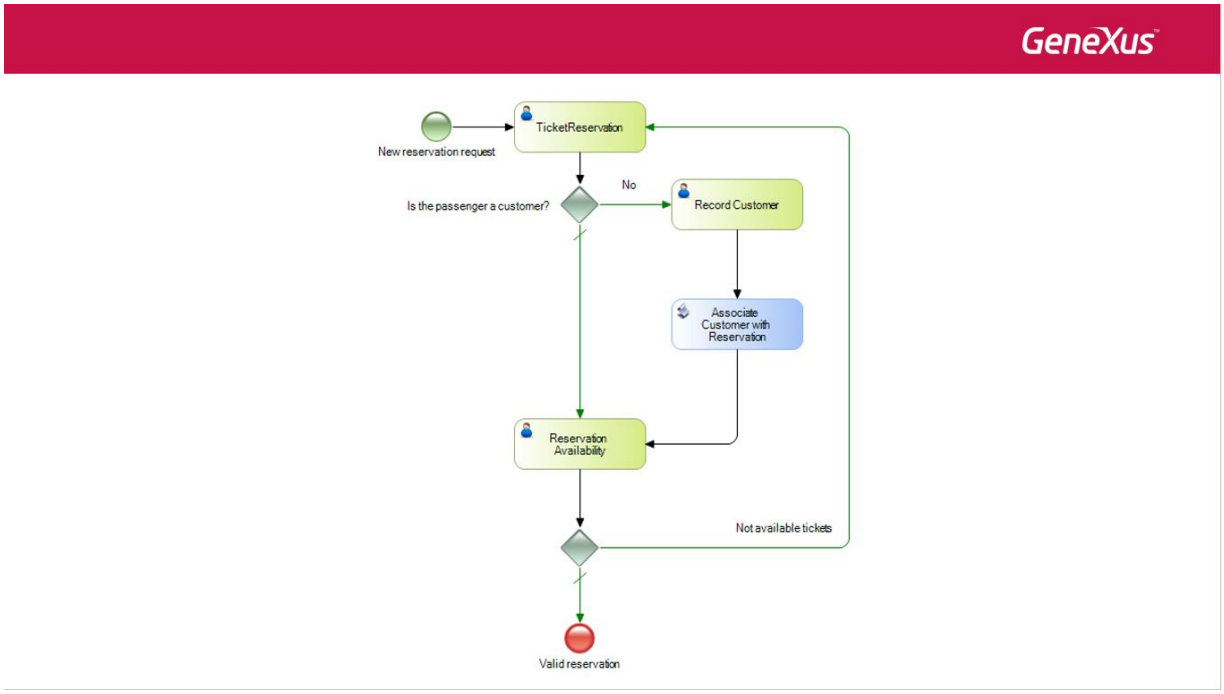
# Running a business process on a mobile device

So far, we have associated GeneXus objects executed on a web page to the interactive tasks of process diagrams.




It is also possible to associate GeneXus objects executed on a smart device, to user tasks. For example, suppose that we want to execute the process of entering a reservation at the travel agency from a mobile device.

We will simplify the ticket reservation process we defined previously. So, we remove the validation sub-process and certain symbols we will not be using. To specify the reservation's availability we will use the Reservation transaction and its ReservationAvailable attribute.

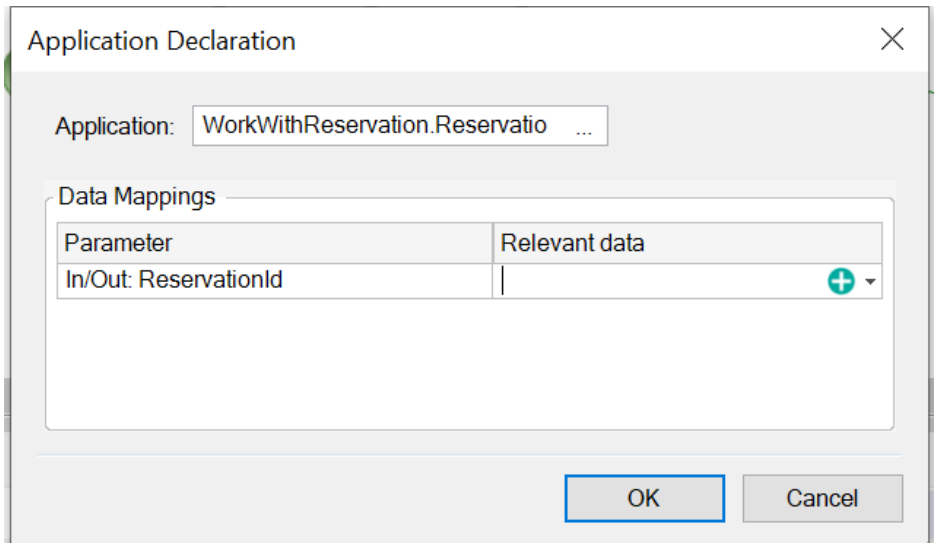



We then select the TicketReservation task, and, for this example, we delete the value of the Roles property

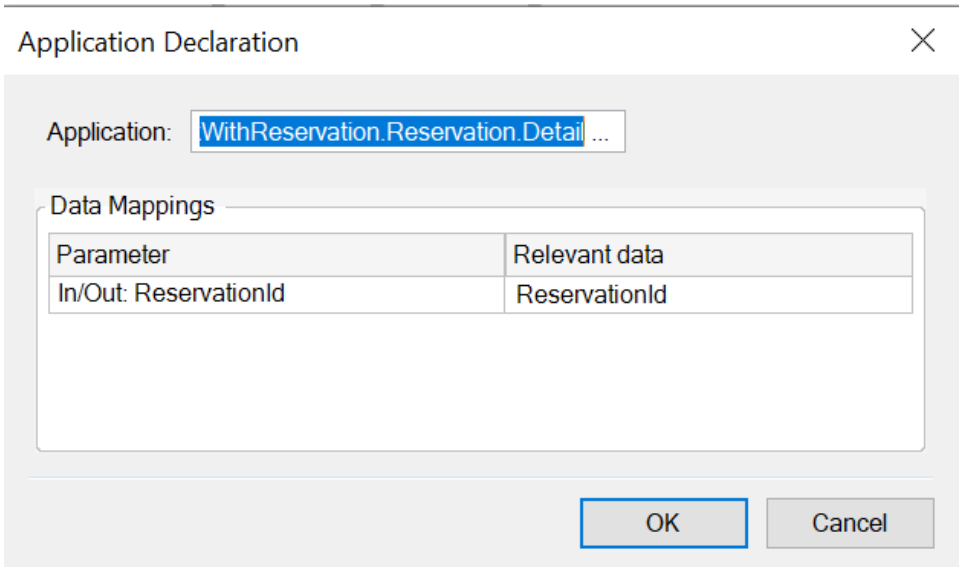
leaving it empty. And we do the same with the None Start Event.
The first thing we will do now is apply the Work With pattern to the Reservation and Customer transactions that are involved in the process. To do that we go to Folder View, select both transactions, right click and choose Apply Pattern and Work With .

We see that, under the node of the Customer transaction is the SD object: WorkWithCustomer, and under the Reservation transaction is the SD object: WorkWithReservation.

Now we will associate the SD application generated by the pattern with the TicketReservation task. To do so we go to the task's properties and in the Object property we press the button and select WorkWithReservation.




We press tab and associate the relevant data ReservationId.



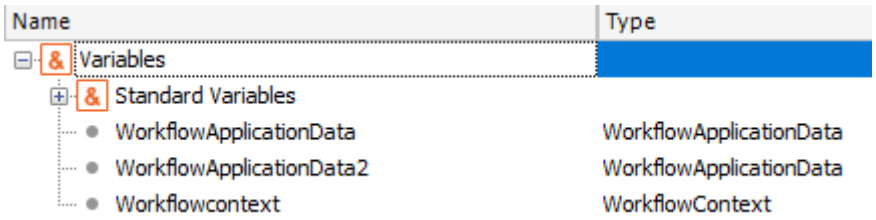

We now press OK.
Something very important to consider when associating SD objects is that **relevant data is not mapped automatically between the diagram and the SD object.** To achieve this association we must have a procedure.

We open the procedure object *ReservationMapRelevantData* that we created previously.
In the rules section we will see that we defined a Parm rule that receives the reservation and

customer identifiers in the ReservationId and CustomerId attributes respectively.

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

The following variables are also defined:




And in the source, using the Workflow API, we obtain the relevant data ReservationId by its name and associate it to the ReservationId attribute we received by parameter.
We do the same with relevant data CustomerId and the CustomerId attribute.

```
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
```

We must not forget to include the Commit when working with Workflow data types.

As we ask for the isNull() value of CustomerId, we can see that we added the following rule to the Reservation transaction:

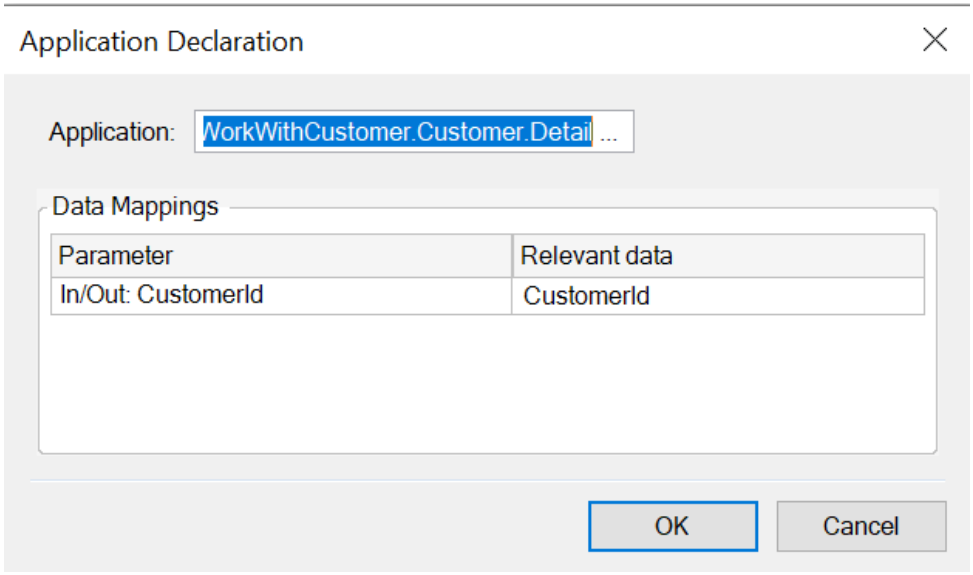
```
6 CustomerId.SetNull() If CustomerId.IsEmpty();
```

Now we open the WorkWithReservation object, then go to its Detail / General section and modify the Save event adding the invocation to the procedure we just created.

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

This will allow that, when we insert a new reservation from the SD application, all the corresponding relevant data will be mapped.

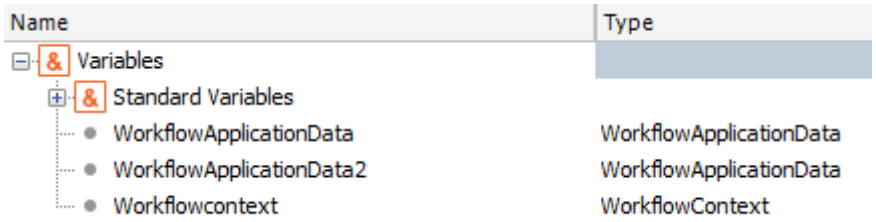
Next, we associate the RecordCustomer task to the WorkWithDevicesCustomer application and assign the relevant data CustomerId:




As it happens with the reservations, we must have a procedure to associate the CustomerId relevant data to the CustomerId parameter of the Customer transaction.
To this end, we have created the procedure called *CustomerMapRelevantData*. Let's open it...
In the rules section we see the Parm rule that receives the customer identifier in the CustomerId attribute:

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

We see the defined variables of the Workflow data types:




And the source implemented, where we obtain the CustomerId relevant data by its name and associate it with the CustomerId attribute we receive by parameter.

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

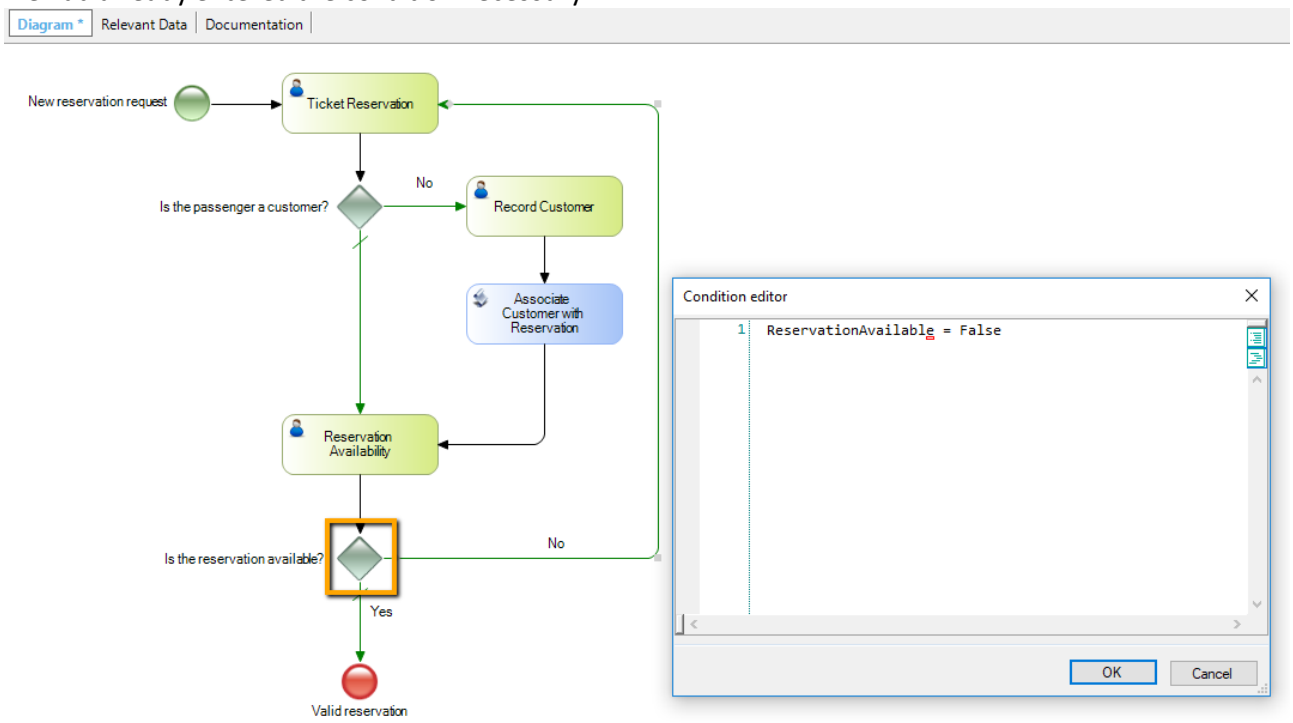
We now go to the WorkWithCustomer object in the Detail / General section, and modify the Save event adding the invocation to the CustomerMapRelevantData procedure.

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

In our diagram, we will assign the ReservationAvailability task to the WorkWithReservations object, similarly to what we did before with the Reservation task.

If the reservation is available, then the process ends. Otherwise, we must enter a new reservation. This assessment is done with the exclusive Gateway.

When we double click on the connector that links the Gateway to the TicketReservation task, we see that we had already entered the condition necessary.




In order to execute the process diagram we built, we must import and set up the **GXflow client for Smart Devices**. Here, we have done it already, but the details may be found at the following link shown on screen:

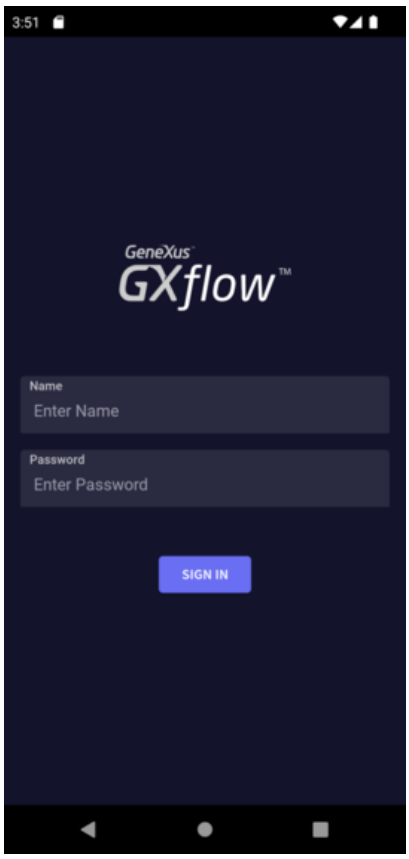
*HowTo: Configuring GXflow Client For Smart Devices*

Once the GXflow client for SD has been imported and set up, we need an invocation to each SD object used in our process diagram, so we must add the following code to the WorkflowSDClient dashboard:

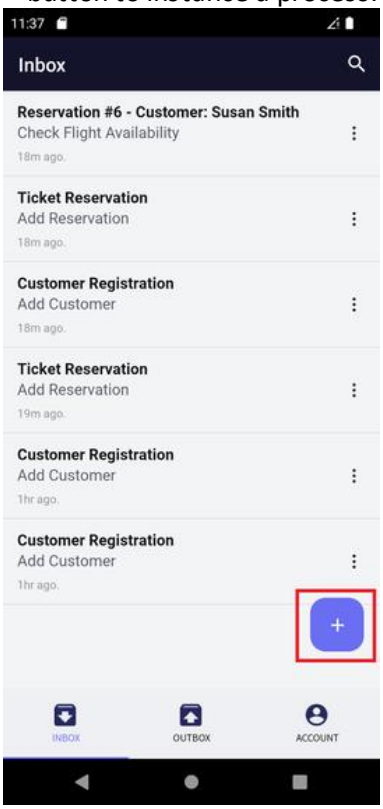
```
Event 'DummyCalls'
        WorkWithDevicesReservation.Reservation.Detail(1)
        WorkWithDevicesReservation.Reservation.List()
        WorkWithDevicesCustomer.Customer.Detail(1)
        WorkWithDevicesCustomer.Customer.List()
EndEvent
```

Now we're ready to execute our ticket reservation process on a mobile platform. So, we first do a Build All, …and then press F5.

We can see that the Android emulator was executed automatically, showing the login screen:

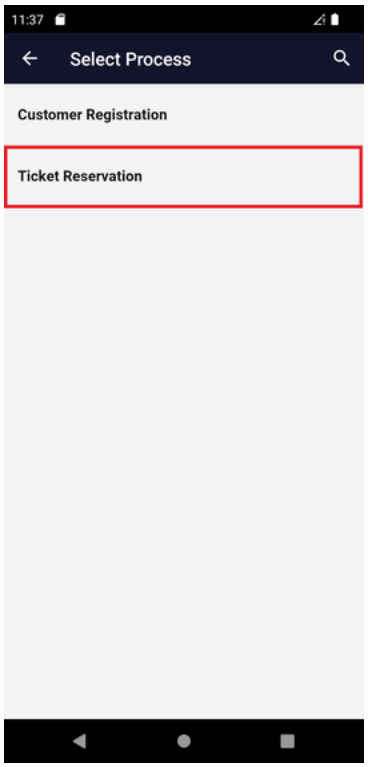



For our login we will use the workflow administrator user, so we enter user: WFADMINISTRATOR and use it also as password.
Upon entering, we view a screen showing the input and output trays We select the input tray and press the '+' button to instance a process.

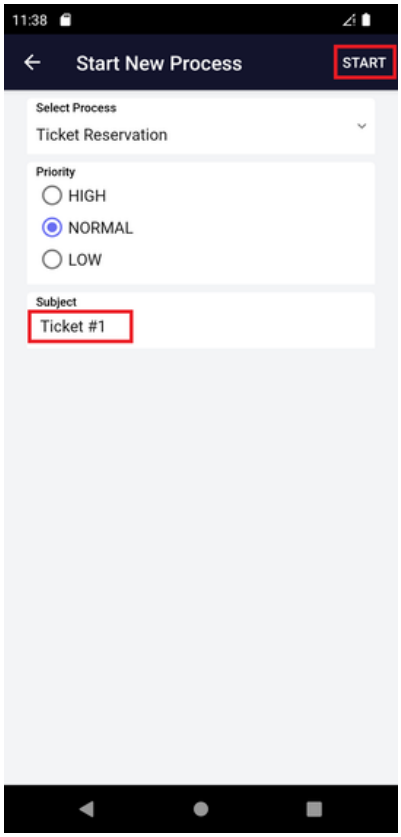



.

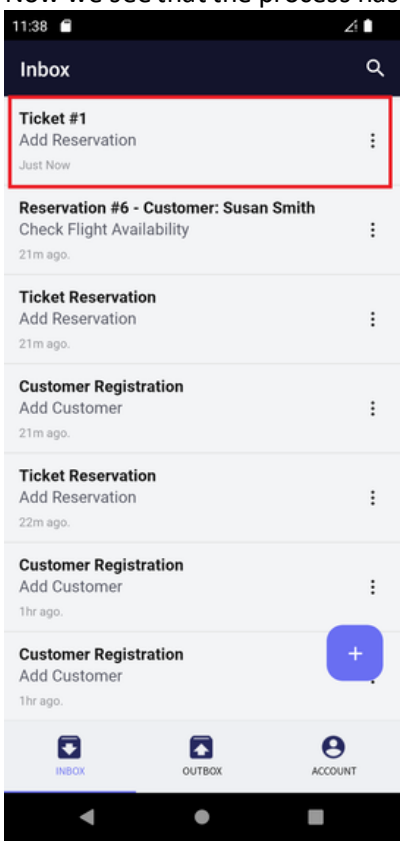
Now we select the FlightTicketReservationSD process.



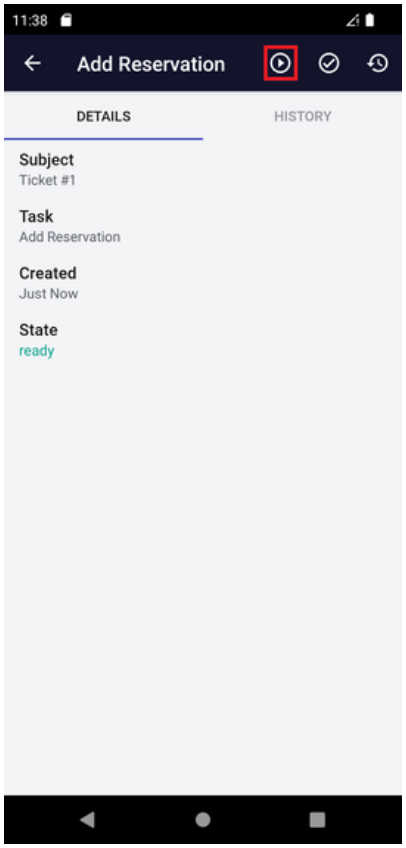

And we will see that the process window opens up. We press the Start button to start the process.



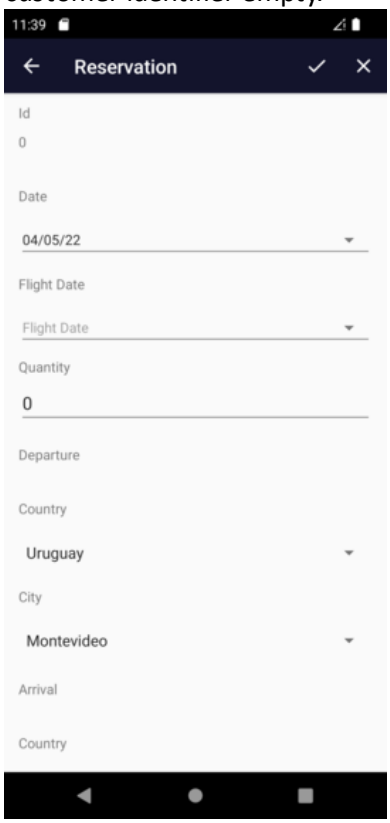
Now we see that the process has been started, and we have the TicketReservation task pending execution.



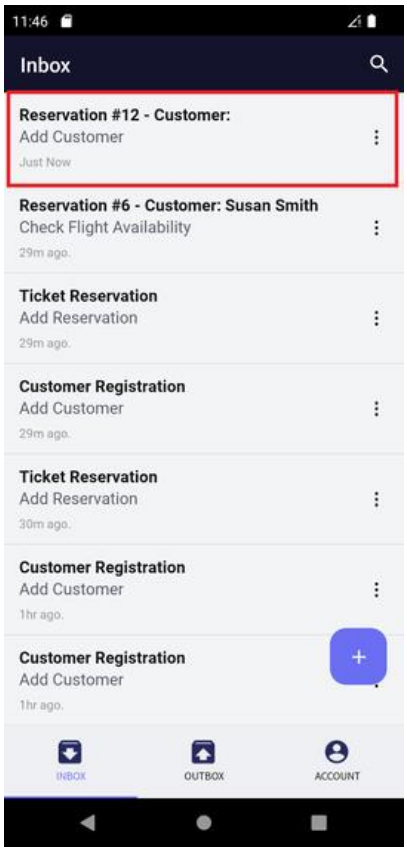

We click on the task, …and press the arrow button to start it.



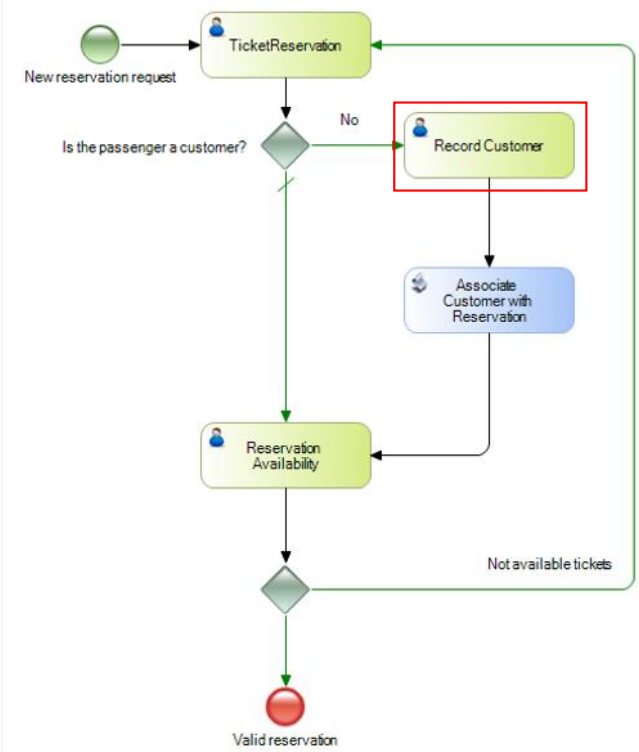
We can see that the SD object to work with reservations is opened, for us to enter a reservation. We will enter a new reservation and leave the ID unspecified because it is autonumbered; and we leave the customer identifier empty.



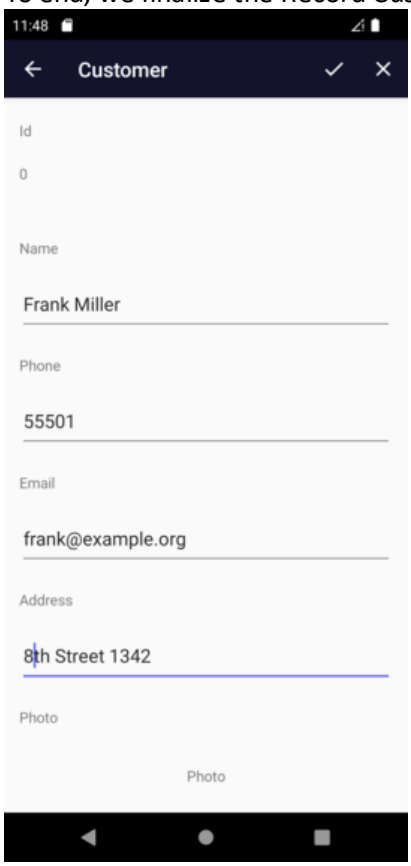

Now we press the Confirm button and to end the task, we press Complete. We will see the input tray open up, so now the task pending execution is RecordCustomer.



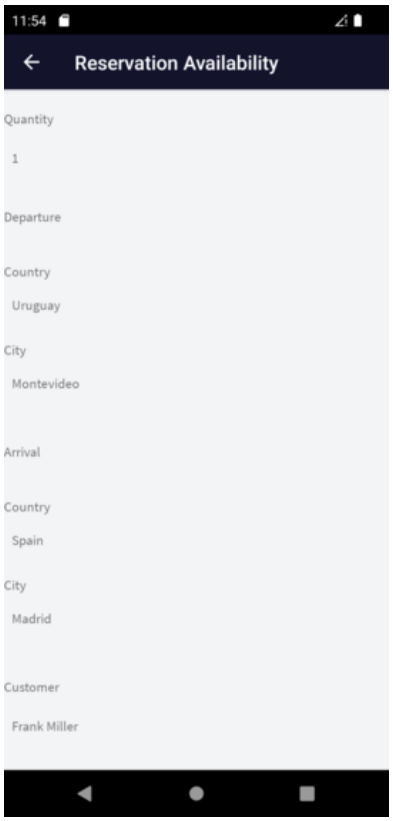
Because we did not enter the customer, the workflow engine assessed the condition of the exclusive Gateway “Is the passenger a customer?”, and determined that the following task will be RecordCustomer, which will invoke the SD object Work with Customers, for us to enter the customer.




We now execute the Record Customer task, …enter the customer data, and press Confirm.
To end, we finalize the Record Customer task.

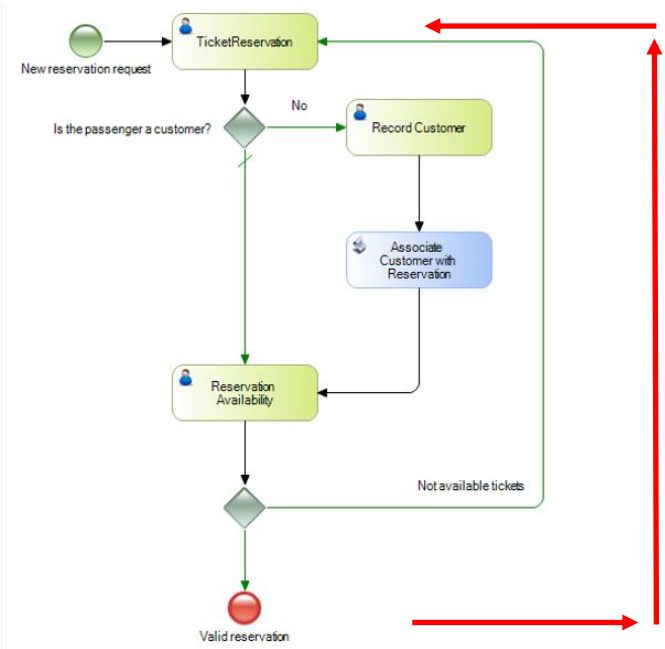


The next task that appears pending is ReservationAvailability. We click on it to open the task,
… and go on to execute it. Note that the customer was successfully assigned to the reservation, because the procedure associated with the batch task AssociateCustomerToReservation was executed correctly.

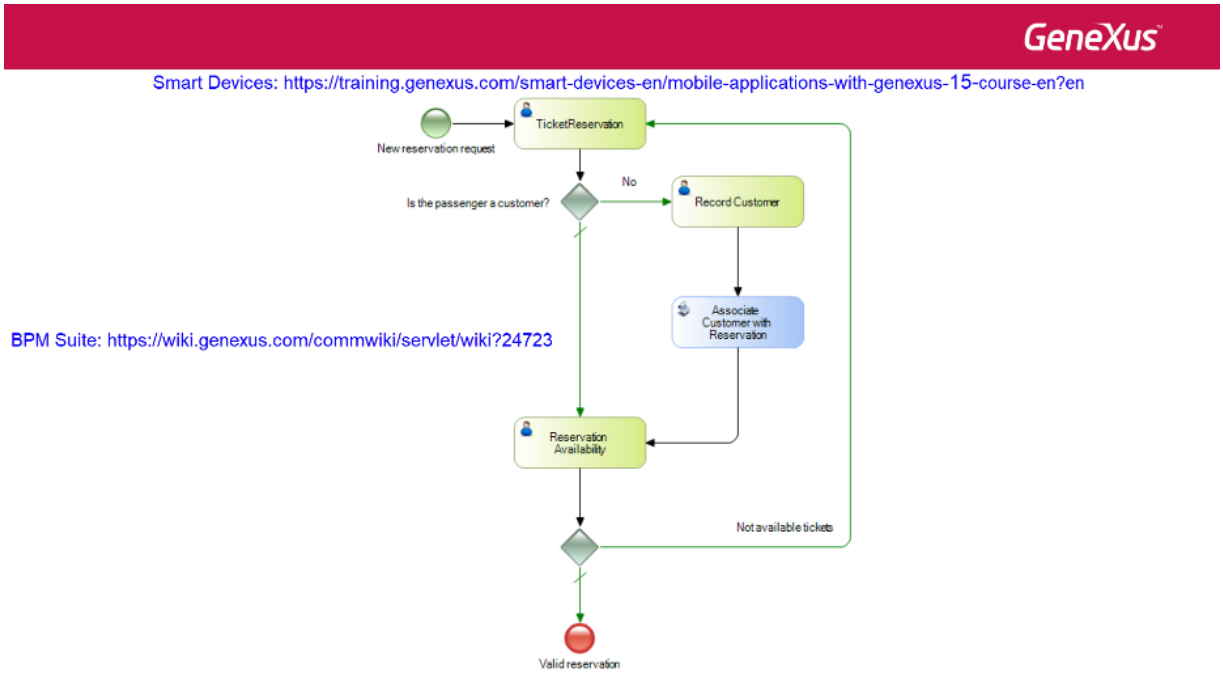



We will indicate with a checkmark that the reservation is available, and then press Confirm.
To end, we finalize the ReservationAvailability task.

Now the input tray does not show any other pending tasks, which means that the execution of the travel agency's reservation process has been completed.
If we had indicated with a checkmark that the reservation was not available, then the SD object WorkWithReservation would have been executed again for us to enter a new reservation.




Here, we have seen how to execute a business process diagram on a mobile device.
Specifically, we have associated the objects generated by the pattern Work With for Smart Devices to the tasks. But we could have also associated a smart device object we created, such as, for example, a panel for SD.
In this case, we generated the application for smart devices on Android, and executed it using an emulator. However, it is possible to prototype on a physical device and generate applications for other platforms such as iOS devices like the iPad or iPhone. To learn more about Smart Device applications visit the link shown on screen:




For further possibilities of the GeneXus BPM Suite, visit the wiki link.