OutSystems documentation

Outystems Definition:

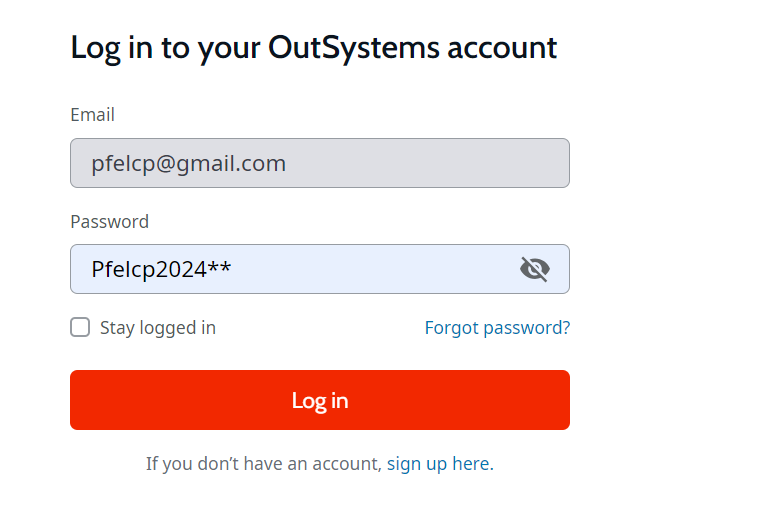
Outsystems is a low-code platform, which means it allow developers to create application with minimal hand coding by using visual development tools and pre-built components.

Email:

Username: [pfelcp@gmail.com](mailto:pfelcp@gmail.com)

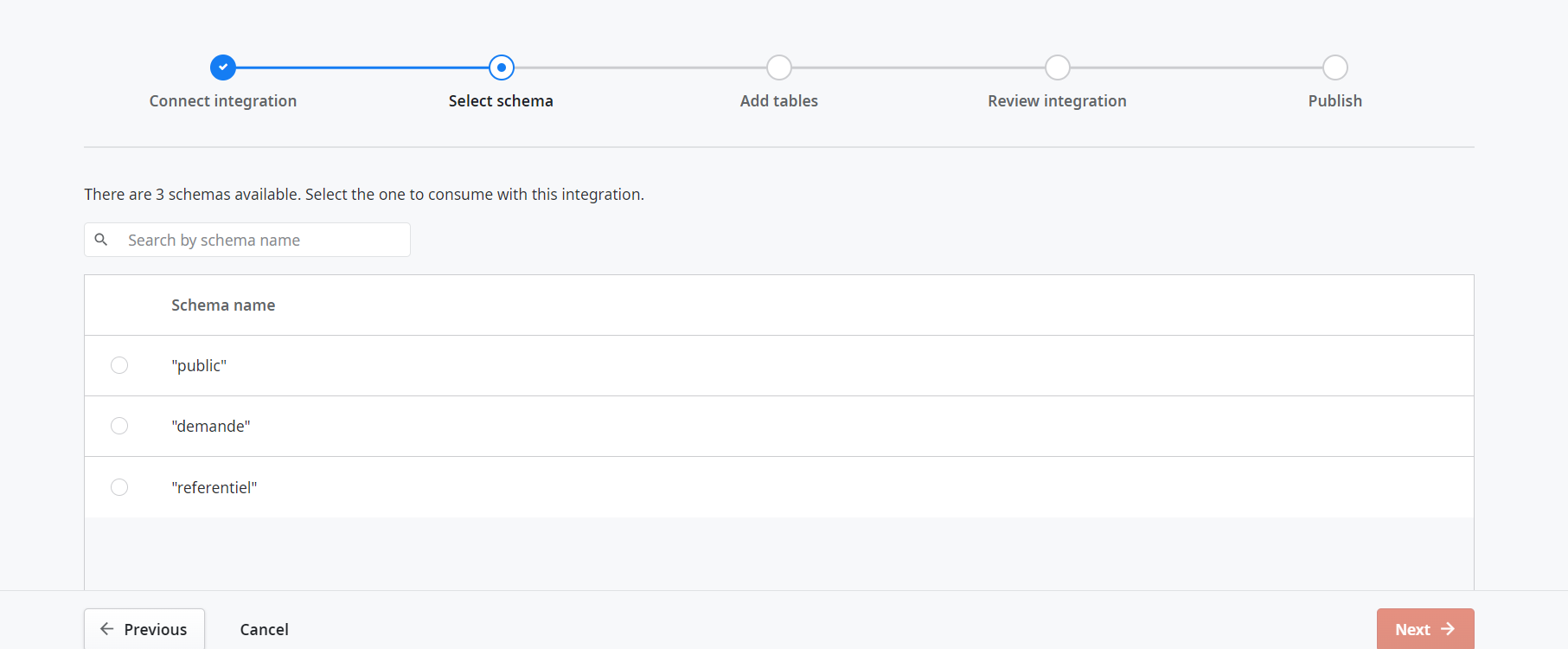
Password: pfelcp2024

Outsystems Account:

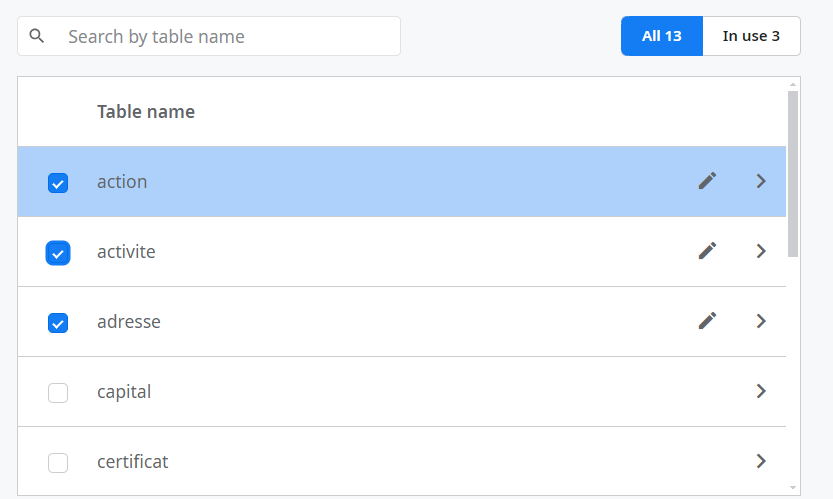


Integration with external DB (Postgres):

1. Go Data -> Database, right click, and integrate with external Database.
2. Create connection
3. Select your schema



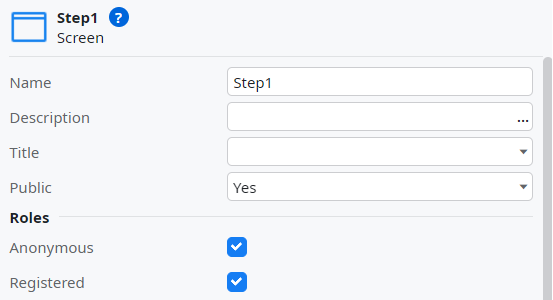
4) Add the tables that you want to integrate in you DB.



1. Review Integration
2. Publish

Integrate Outsystems with Angular:

First you need to put all screen in your out systems app in anonymous roles to avoid the login request.



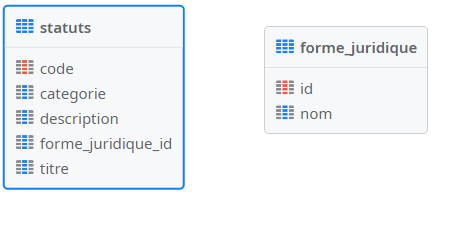
Now for example if we have screen with this url: https://personaljin5krwd.outsystemscloud.com/RNE/DemandeDetail

You put it in iframe and that’s it.



Use case:

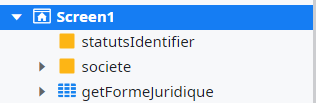
We will implement dependent dropdowns (formeJuridique and TypeStatut) and then perform CRUD operations on the societe object.



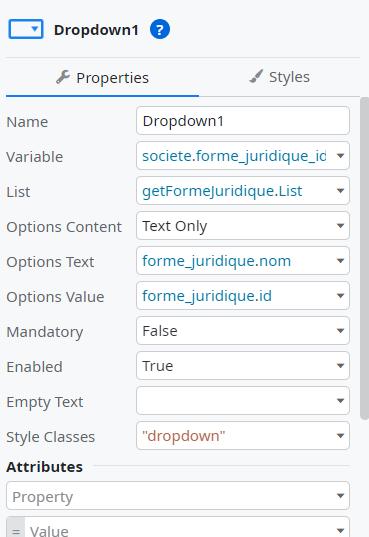
First, we will start with the dependent dropdowns.

* Fetch formeJuridique in dropdown:

After creating your screen, you can fetch the dropdown data from the database. For example, if you have "screen1", right-click on "screen1", select "Fetch data from database", and then choose the "formJuridique" table.



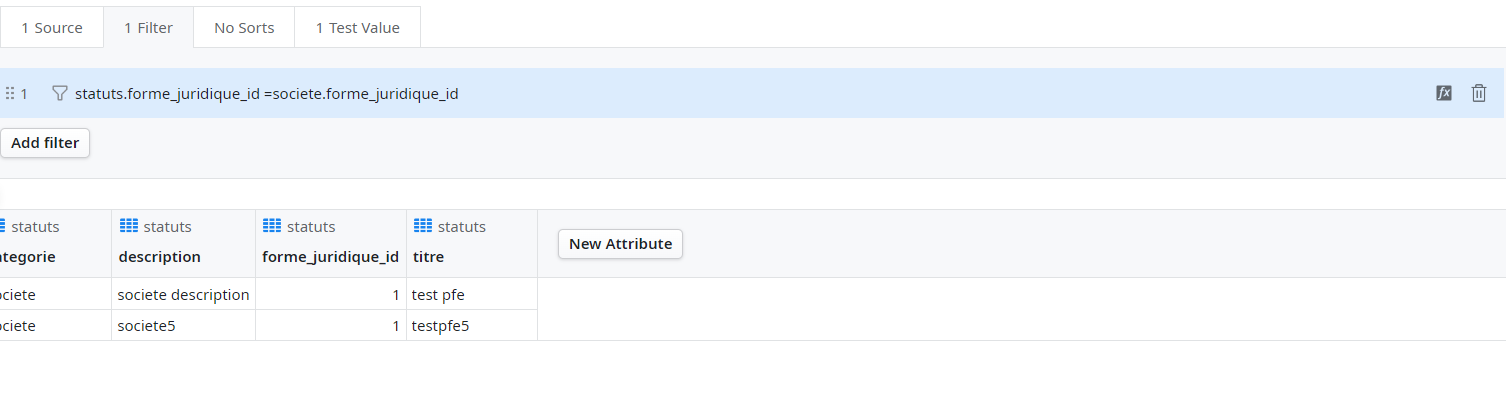
Next, select your dropdown widget, go to its properties, and populate it with the appropriate data.



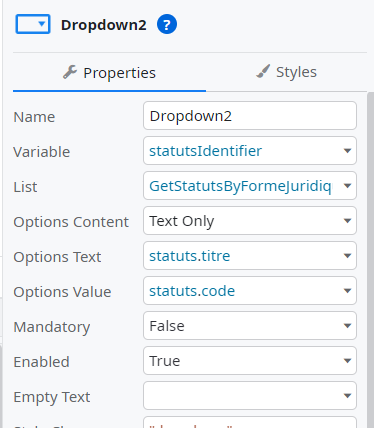
Fetch "Statuts" in the dropdown based on the selected "formeJuridique":

Right click on screen1 and click on fetch data from database and select Statuts

Next, add the attribute that you will use to filter the "Statuts".



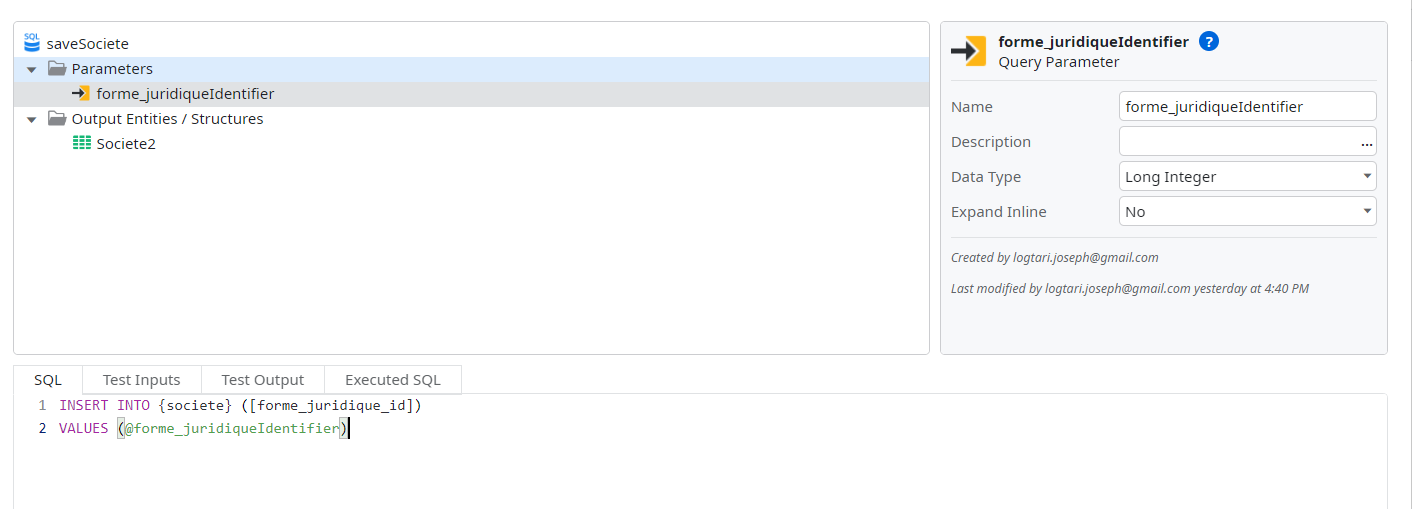
Finally, go to "screen1", import your dropdown widget, and configure the necessary properties.

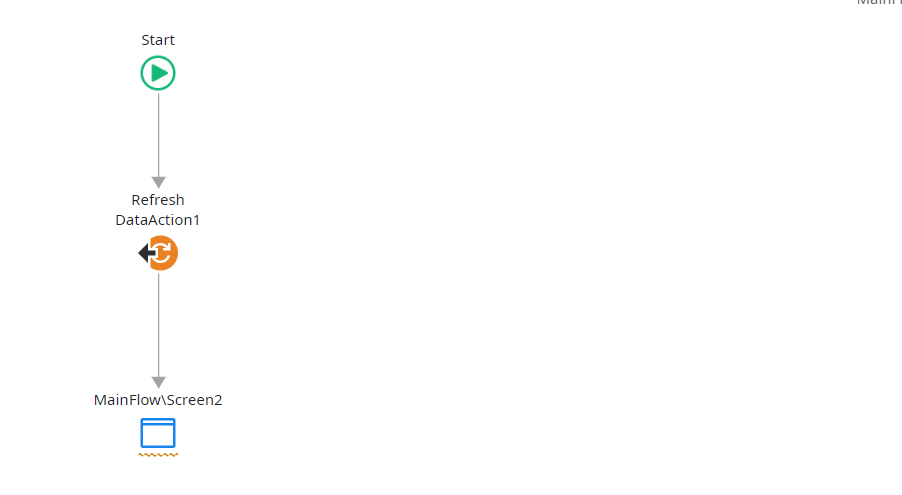


Ok now we will move on to the save of the societe .

Click right click on screen1 and click fetch data from other source.

Drag and drop the SQL functionality, execute your insert query, and then add your parameter and output structure.



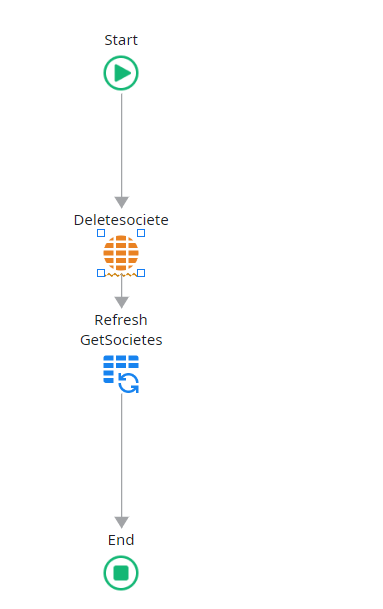
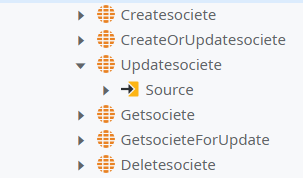
Now add a client action and add your data Action to it. And link the client action to your save button. 

Delete Societe:

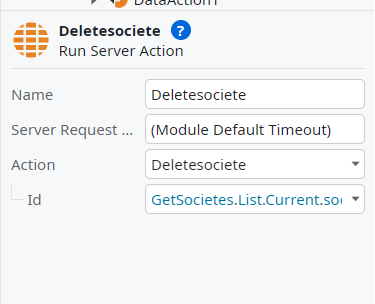
Alright, let's proceed with the delete action in another screen called screen2.

First, fetch the "societe" data we added before. Right-click on screen2, select "Fetch data from database", and choose "getSociete".

Now, drag and drop the "getSociete" action onto screen2. After that, add a delete button and link it to your client action responsible for removing the "societe".

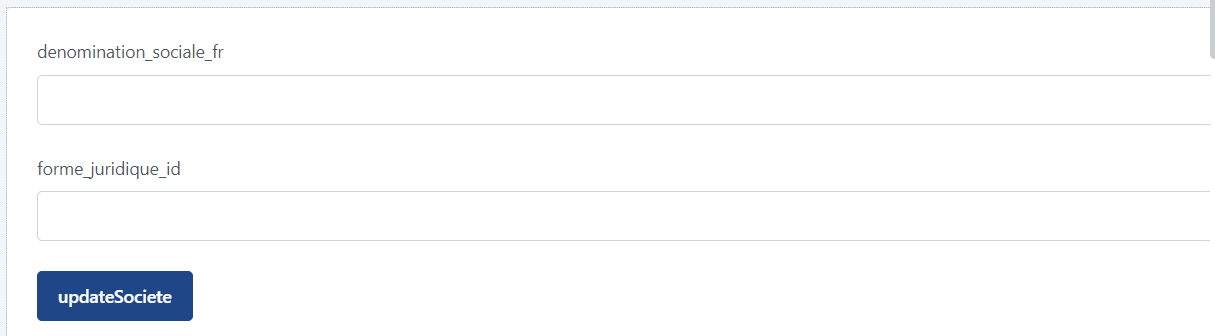
 

Drag and drop your deleteSociete action fo your societe Data into your client action.

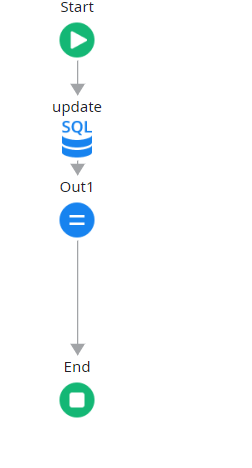


Update Societe:

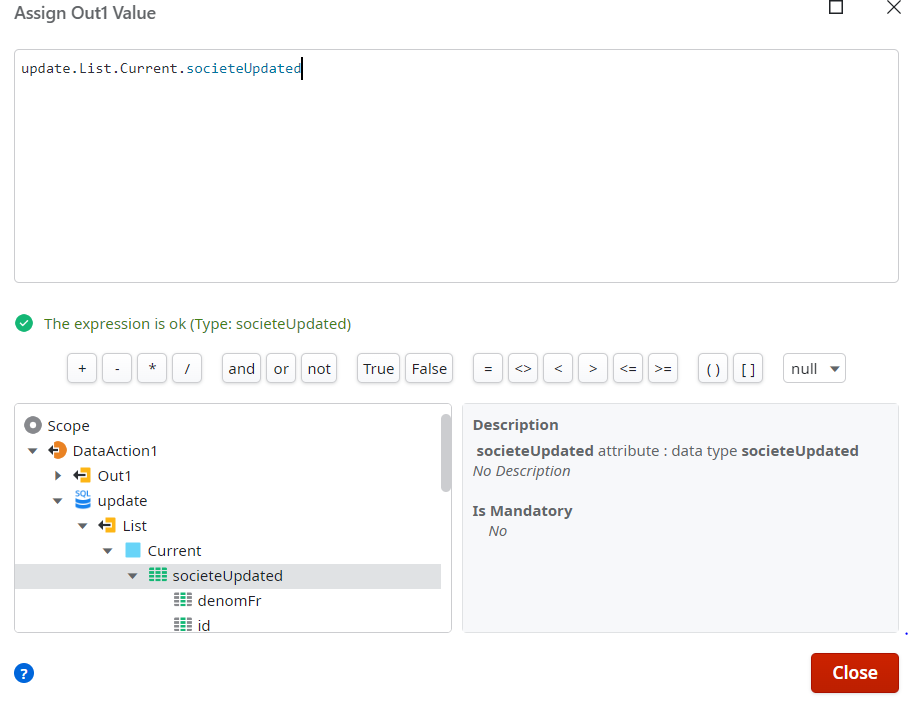
Now we will do the update in another screen called Screen3. In screen3 add societeId as input parameter and fetch data from database and call it for example “getSocieteById” ok now you add a form widget in the screen3 and drag and drop the “getSocieteById” and choose the attribute that you want to fetch it.



Ok now add fetch data from other database and drag and drop the sql functionality and execute your update query as we do in the saveSociete.

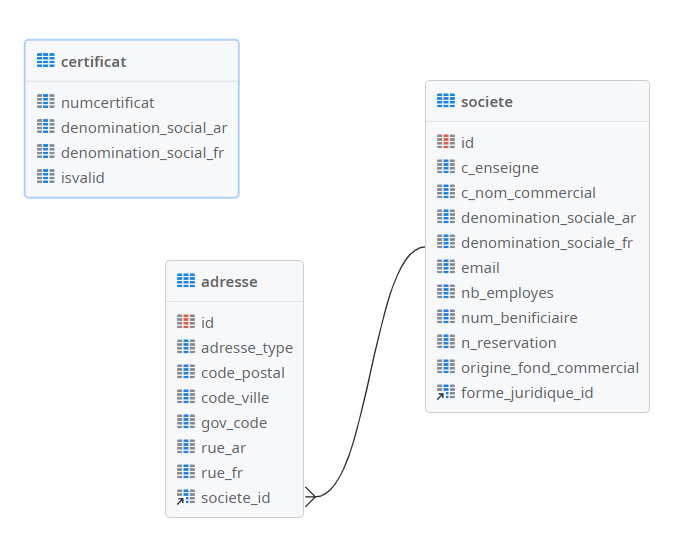
 

Now drag and drop the assign functionality and assign value to out1

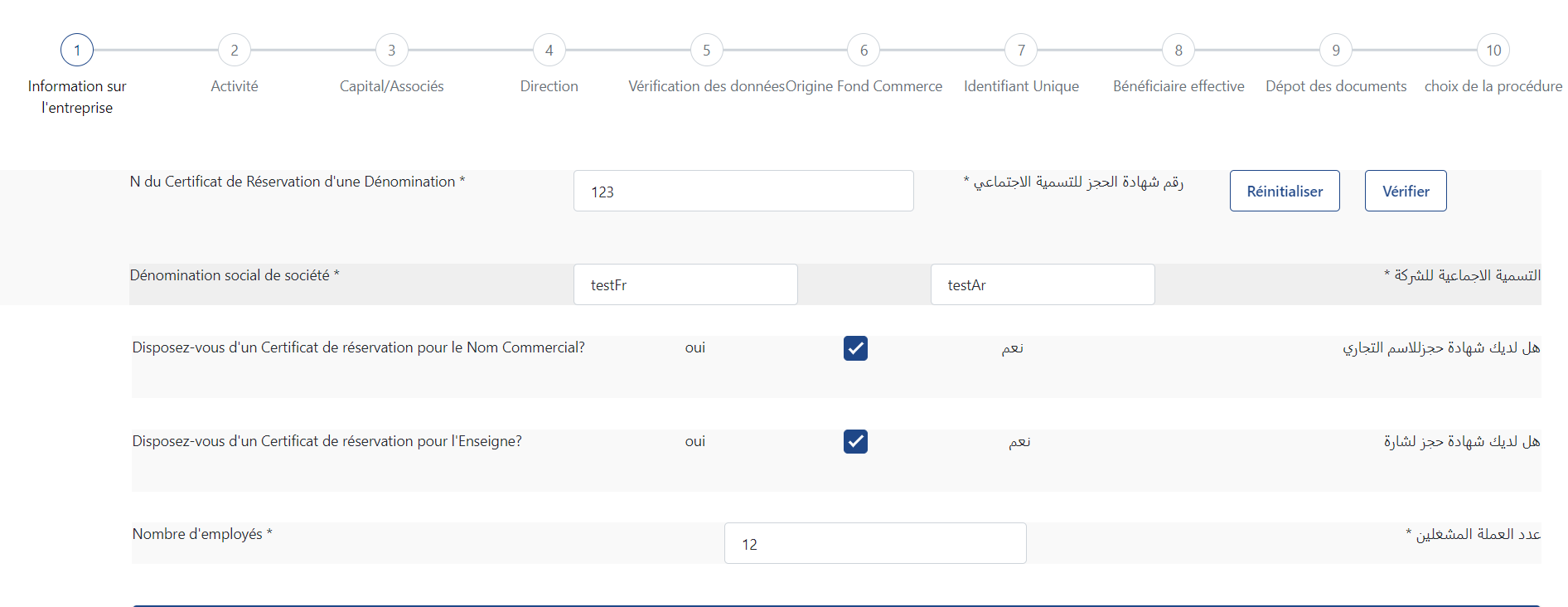


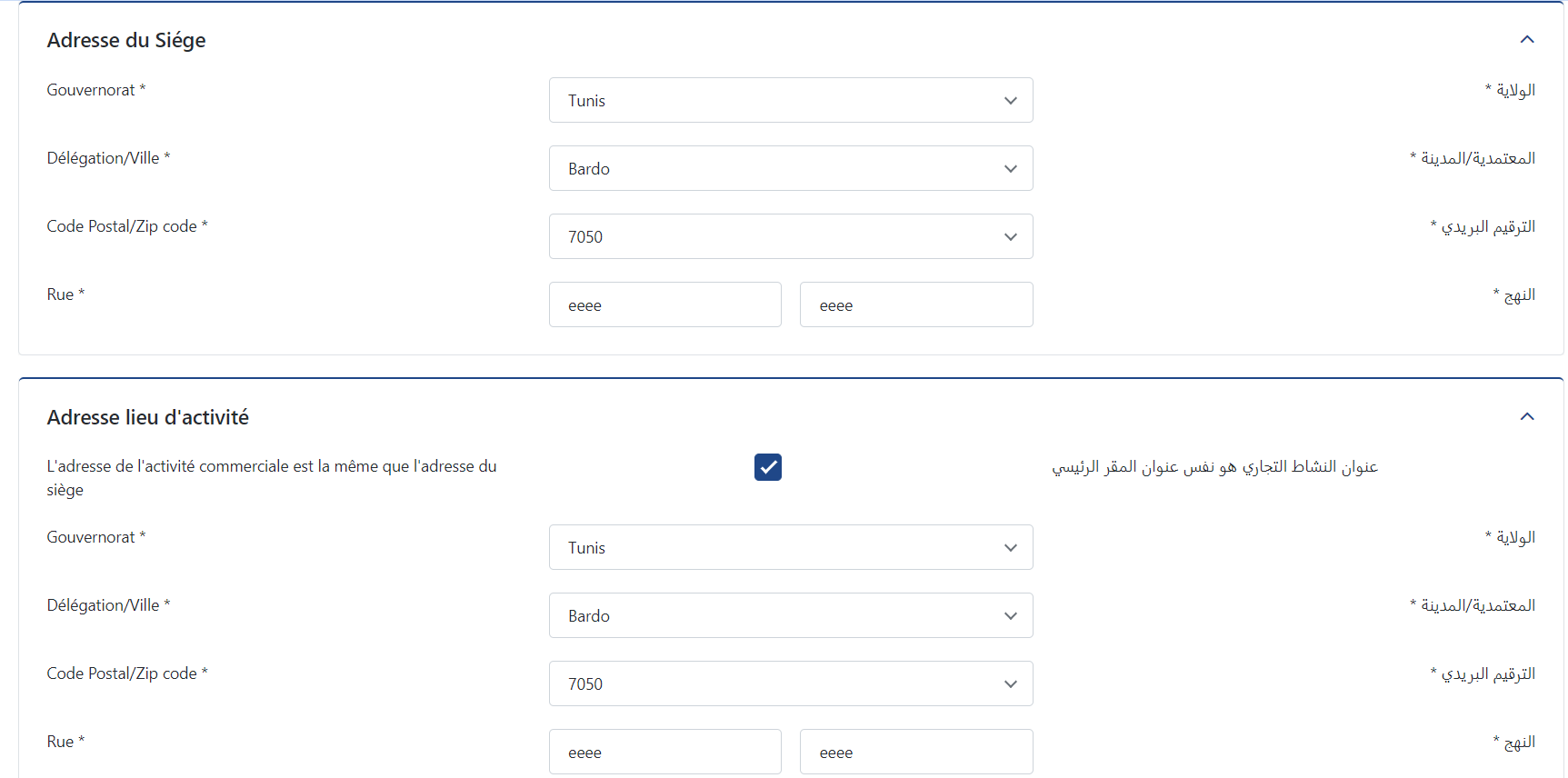
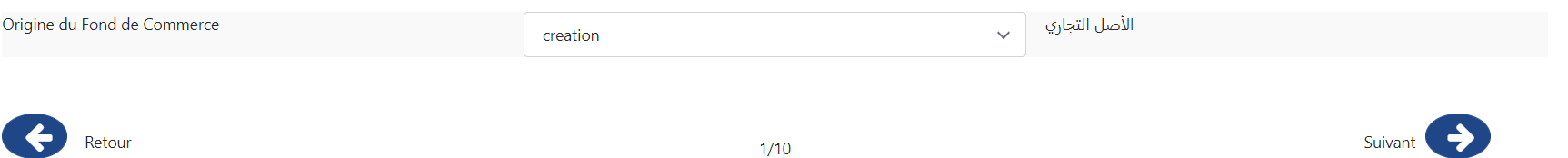
Step1:

useCase:



in Step 1 we will fill out the form below

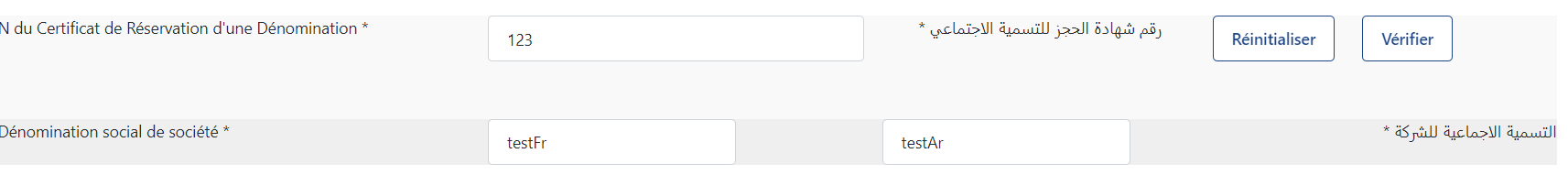


And to do that we will follow these steps :

1. Certificat verification
2. Fill the form with the correct data
3. Click on button Suivant to update in the société table

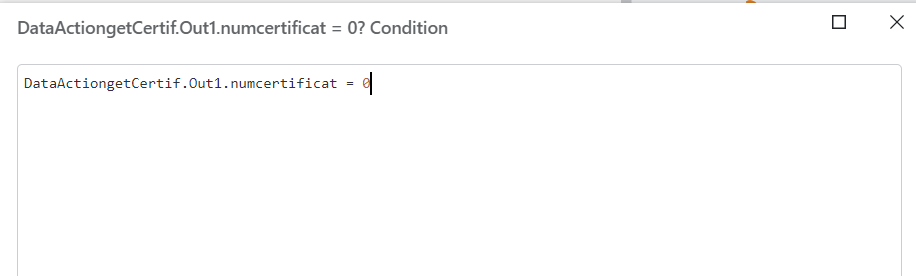
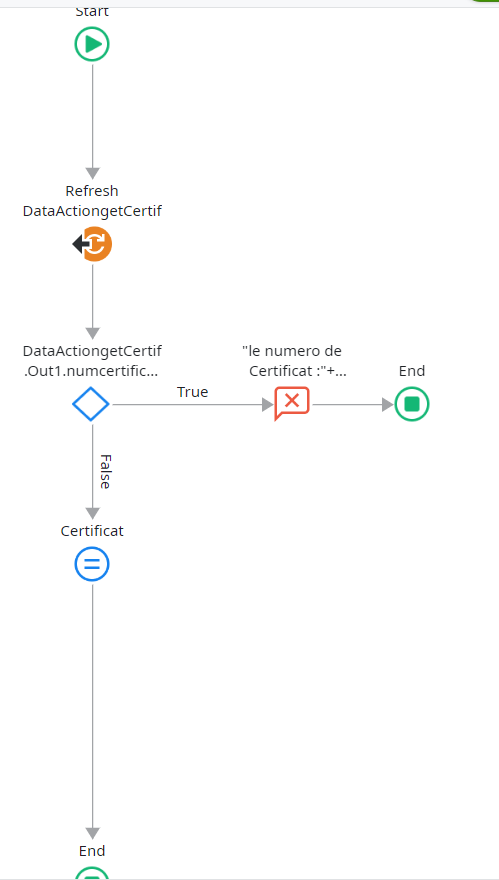
Ok lets start with the first step: Certificat verification:

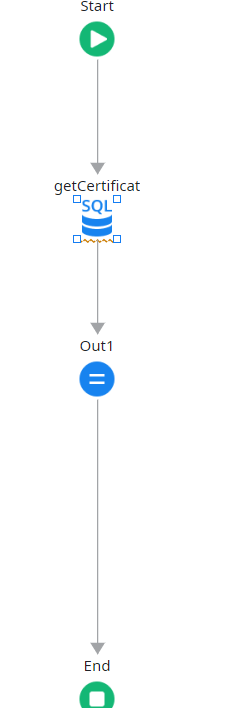
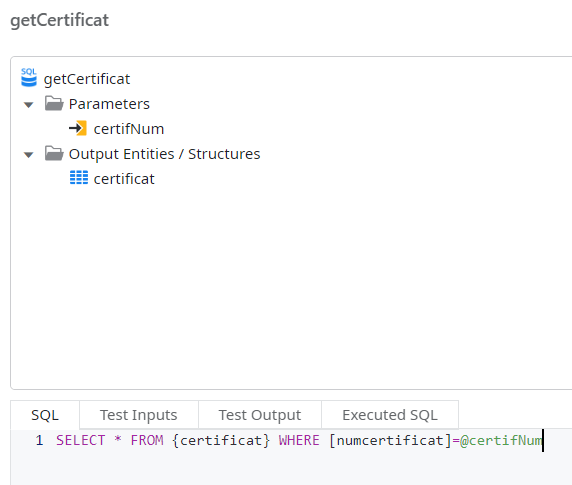


First you should put a variable for your input widget .for example here I put these variables: for certificatNum :certifNum , for denominartionFr : Certificat.denomination\_social\_fr and for denominationAr: Certificat.denomination\_social\_ar.

While the click on the vérifier button we will implement these client action:

In this client action, we retrieve the certificate object based on the certifNum provided in the form. If the certificate exists in the Certificate table, it is assigned to the Certificate object; otherwise, an error message is displayed.





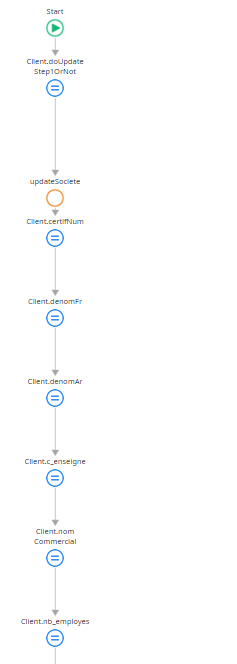
DataActiongetCertif

2) Now let’s move to the next Step: Fill the form

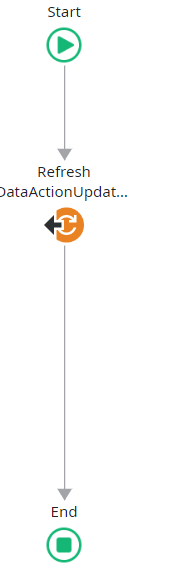
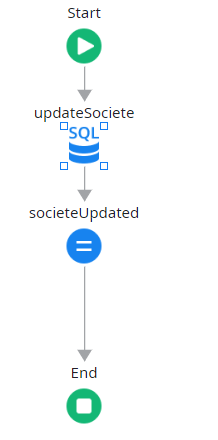
We do the same thing as we did for the certificate: bind your input widget to the variables where you want to store the data.

3)Click on button Suivant to update in the société table:

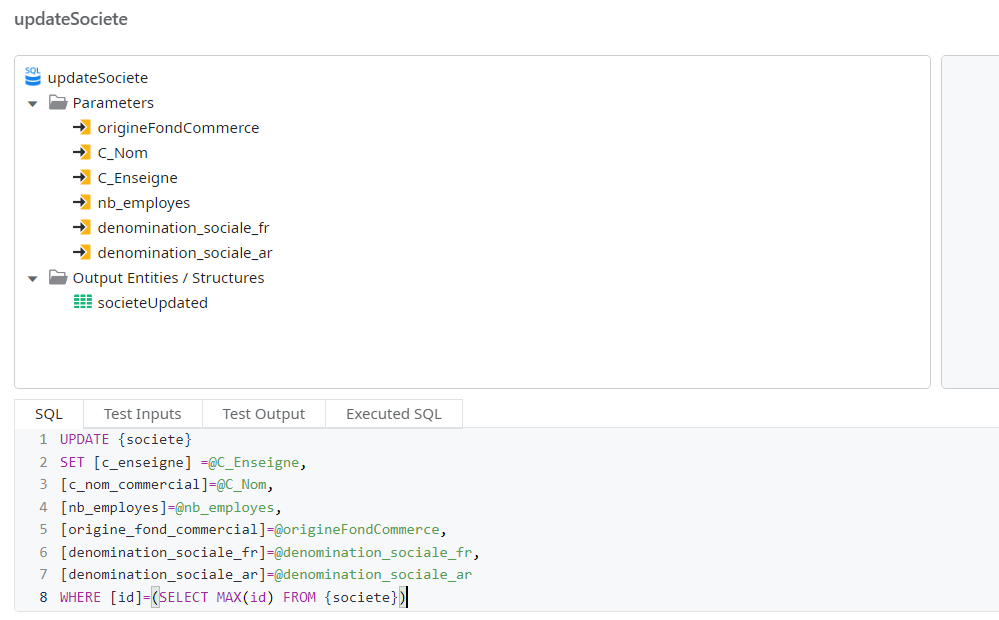
When we click on the Suivant button we implement these action client:



For the updateSociete Action:

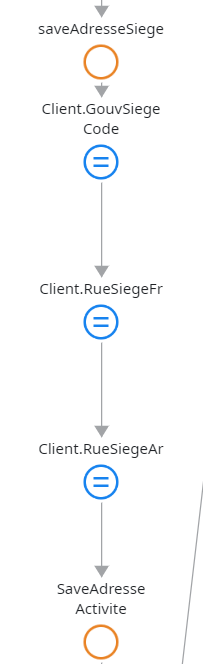
 

DataActionUpdateSociete

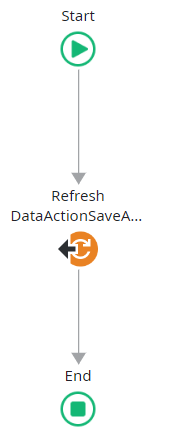
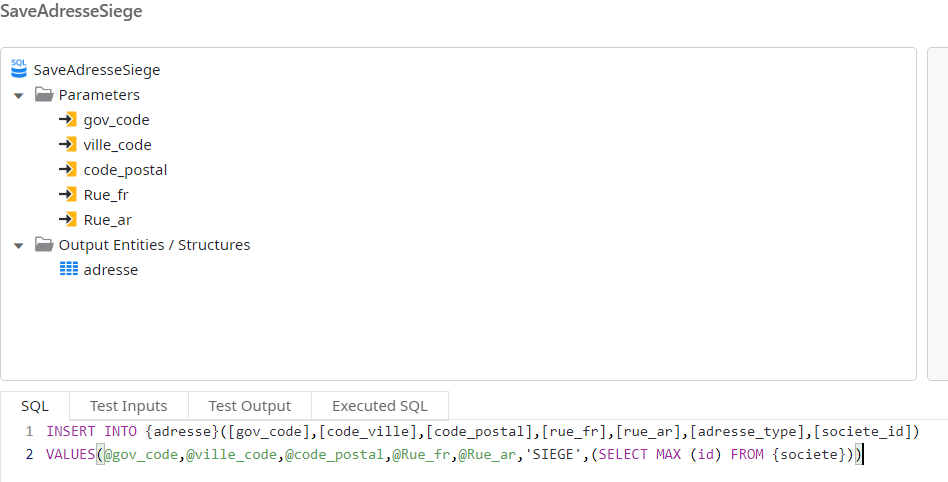


And the output of these query we put it in societeUpdated structure

Now for the save of the adresseSiege and adresseActivite:

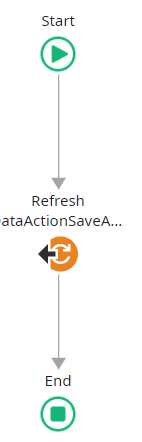


For the saveAdresseSiegeClient:

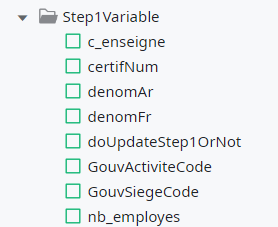
DataActionSaveAdresseSiege

For the saveAdresseActiviteClient:

Before moving to Step 2, if you want to save your data while navigating between pages, you should store all the variables in a client variable. Although you can use input variables, it is not the best practice. Input variables will appear in the URL, which is not secure, and this approach can also slow down the process.

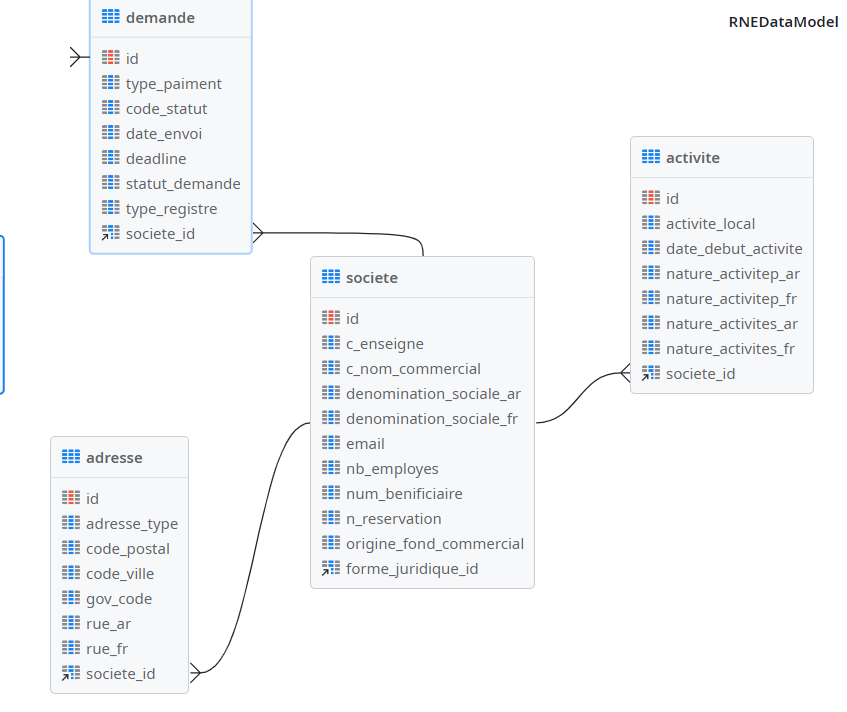
These are the client variables:



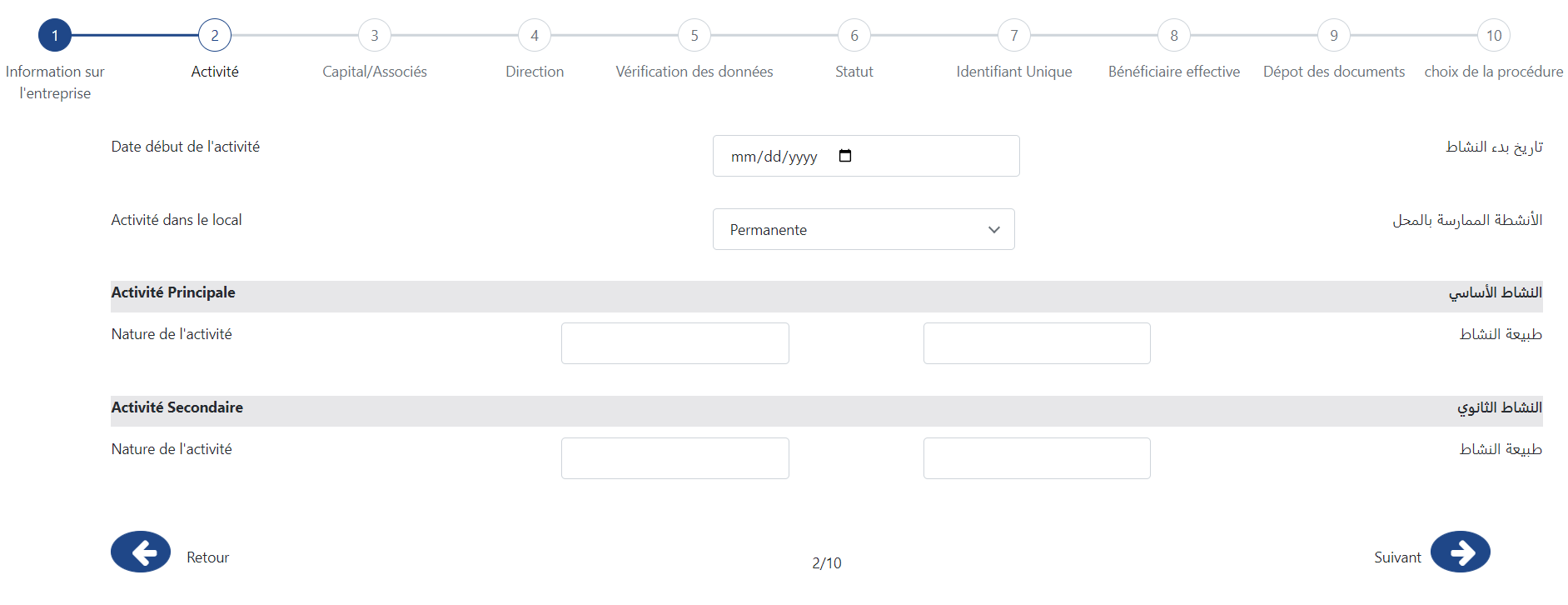
And after save the entity you should bind the variables with their client variable.

And then you call these client variable in OnInitialize event.

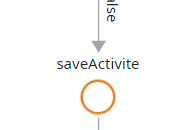
Step2:



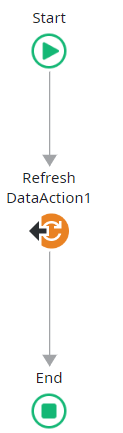
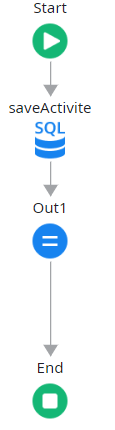
useCase:



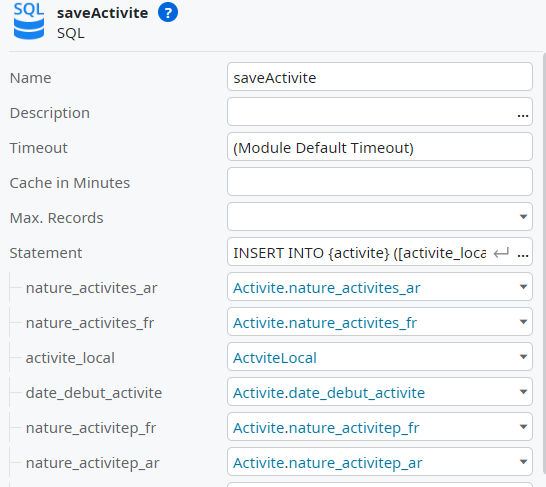
Here same thing as step1 you should bind your widgets with variables that you will passe it the saveActivity query. While clicking on Suivant button.



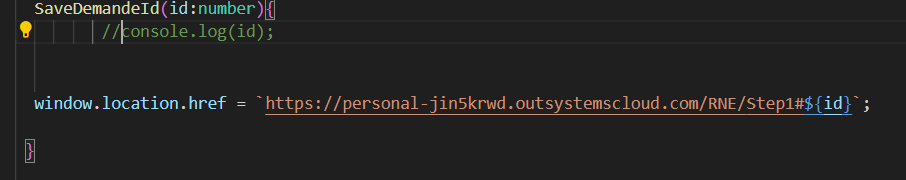
For the SaveActiviteAction:



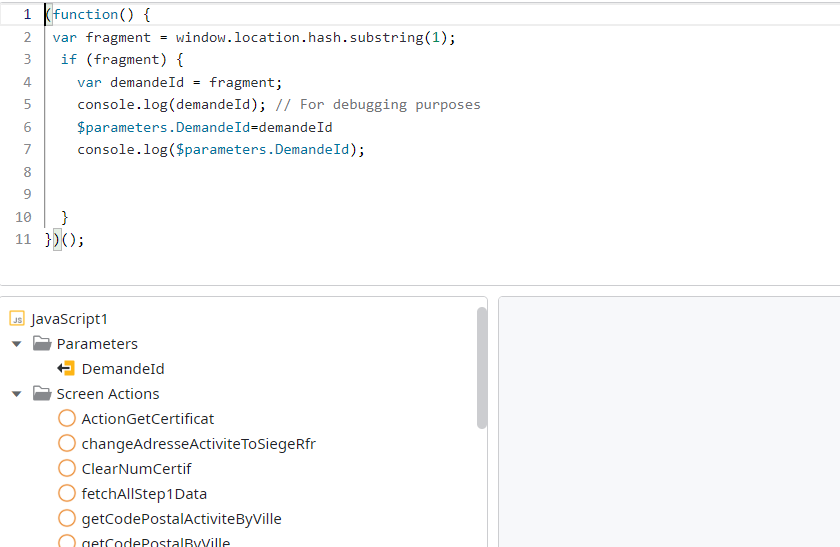


PASS THE DEMANDE ID FROM ANGULAR TO OUTSYSTEMS:



First you pass the id demande in the url of your outsystems app .

And then create a JavaScript action your outsystems app to get this id:

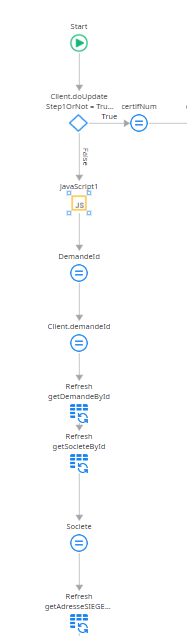


RQ:

You should define an output variable to store the result of your JavaScript inside it. For example here DemandeId . and then you bind the output variable with a local variable as you want.

And then you fetch all the data that you want based on the demandeId and use aggregate to do that and filter based on DemandeId.

After that you call the data that you fetched inside an Initialize event.



Outsystems limitation:

* The applications developed using OutSystems are hosted on the OutSystems cloud, unlike platforms such as Mendix which offer more flexibility in local or on-premises hosting.
* OutSystems does not natively support Git for version control, which can complicate collaborative development.

Conclusion:

These critical limitation do not align with our enterprise policies and operational needs, leading us to seek alternative solutions that better meet our requirements.