Demo Creating a Mobile App.

Developing mobile apps with OutSystems is easy. If you have an Excel file containing your data, you can import it into a database and quickly create a mobile app that enables you to check and manage your data on the go.

To create a mobile app with data that's imported from an Excel file, you need to:

1. Create a database model, and import the data from the Excel file into the database
2. Create a screen that lists the data from the database
3. Synchronize your server database with your local database
4. Implement functionality to synchronize
5. Add a plugin to your application
6. Test the application on your mobile device.

Let's do this! In this example we'll use a sample Excel file with Employee information, and we'll create a simple Contacts mobile app.

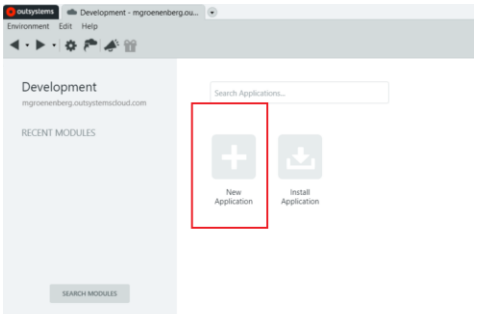
Before you start you have to download two files from a GitHub repository
https://github.com/Synobsys/Demo.

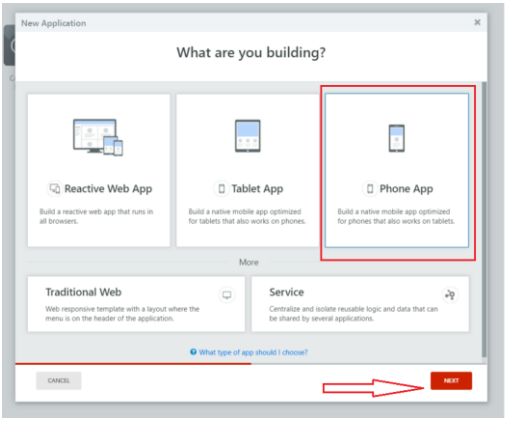
One file is called `Employee.xlsx` the other is an icon called `contacts.png`. Store these files on your PC.

Contents

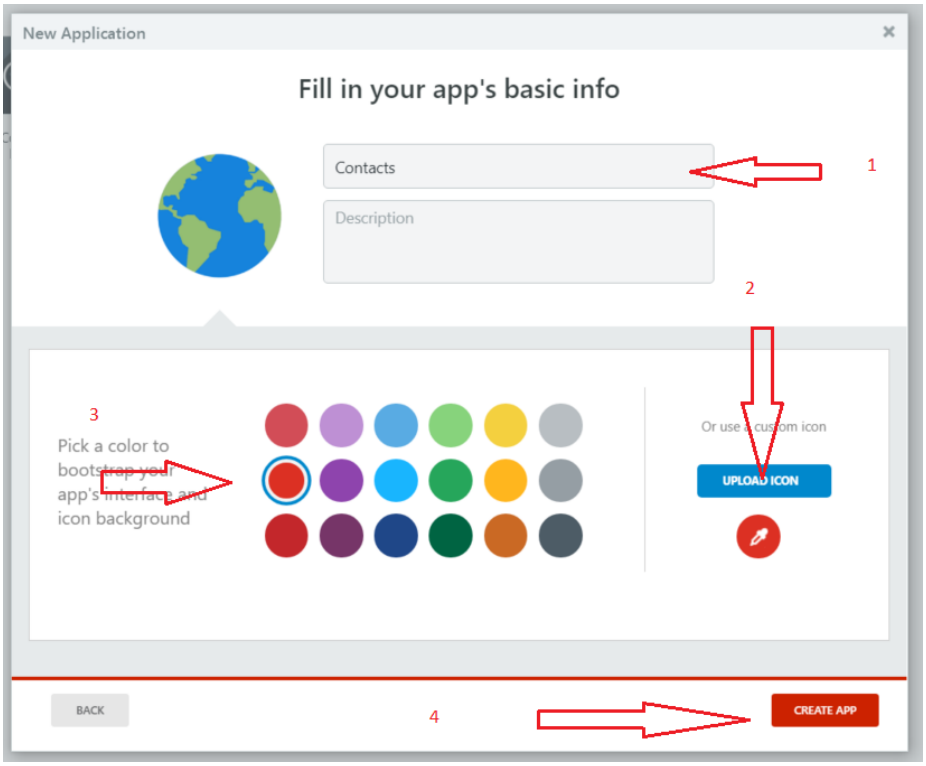
1. Creating your application

In Service Studio, click **New Application**, choose **Phone App**, push the **Next** button

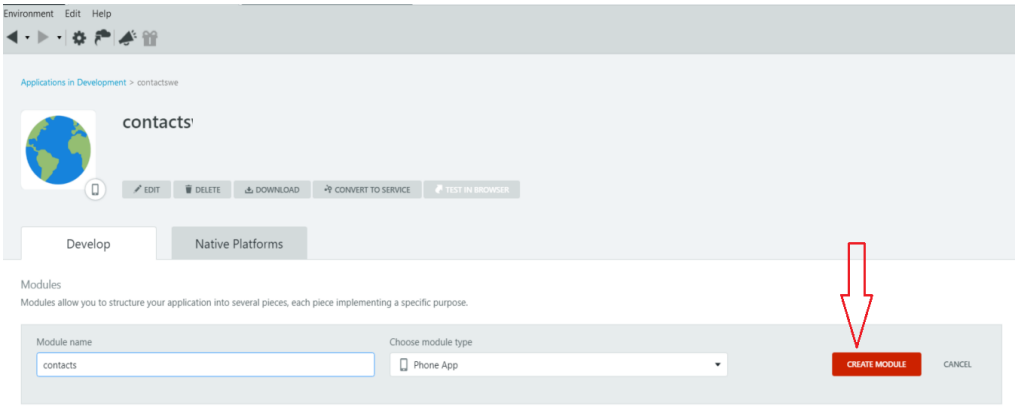




Enter name `Contacts`, Select the Icon you downloaded from the site `contacts.png`. And select a base color for your application. Push the button **Create app.** See below.

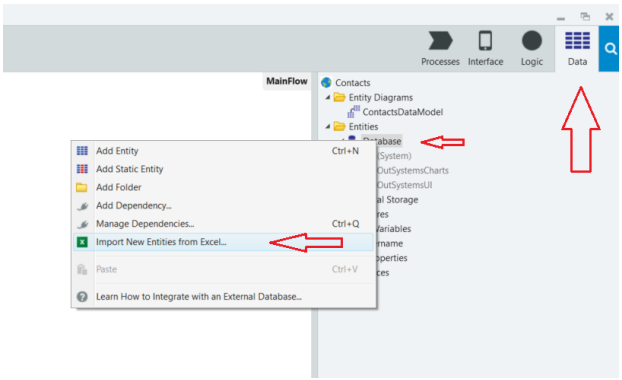


You now have an application. Now Outsystems offers you a screen in where you need to create a Module. The Module name is already filled in. Push the button **CREATE MODULE.**

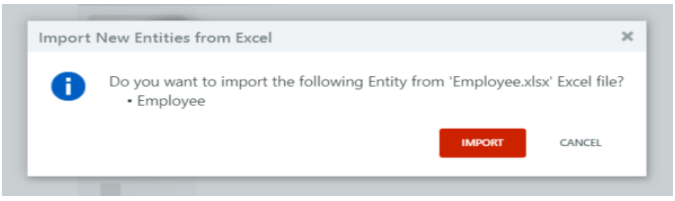


2. Creating the database

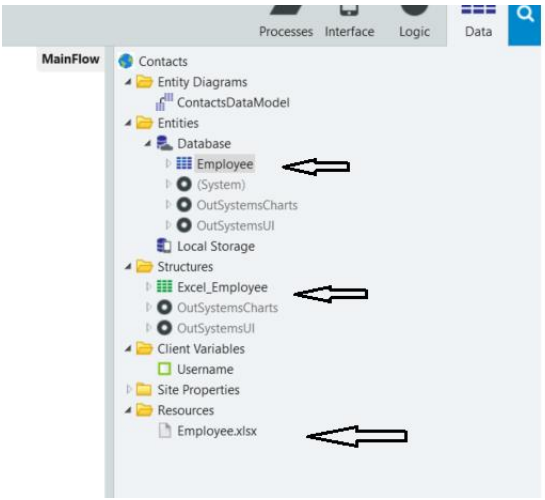
Open the D**atabase tab** in the top right header of the screen. Right click with your mouse on **Database** and select the option **Import New Entities from Excel.**



Select the downloaded file "`Employee.xlsx`" which you downloaded already and open this file. Wait for the confirmation and Import the file.



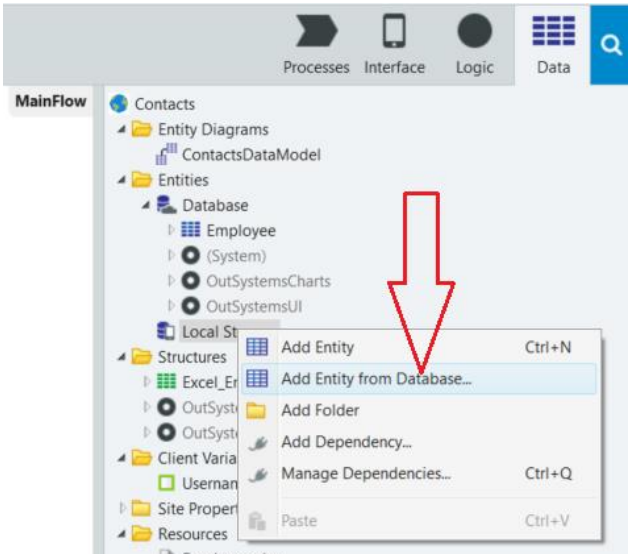
After the import your application looks like this and all definitions are imported by OutSystems.



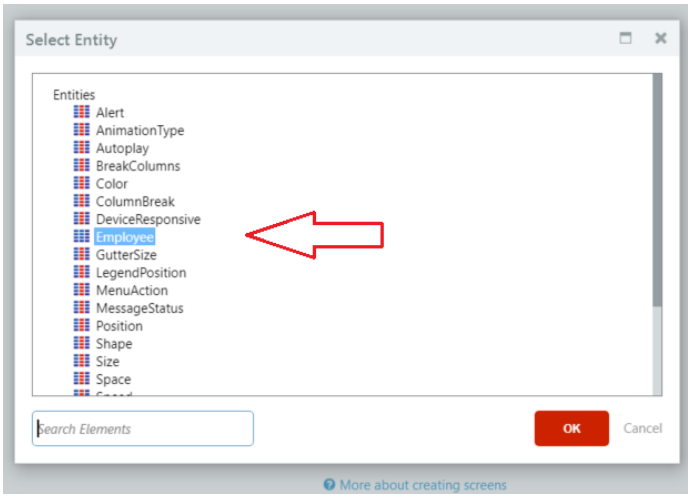
3. Local Storage

Because this file location is on the Server side we also want this file on our local device. All Android- and IOS-devices uses SQL-Light and so it is easy to do. We can use our device also when it is not connected to the Internet, and getting data from local storage is also faster.

Still in the database tab we click right with the mouse on **Local Storage** and select **Add Entity from Database**…

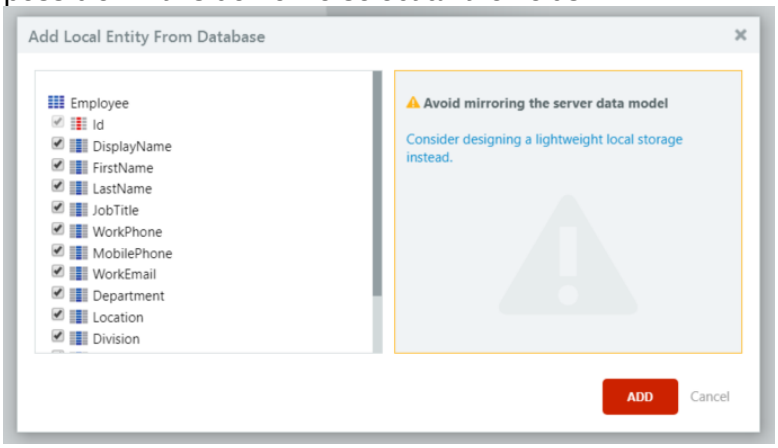


Select the `Employee` database and confirm with **OK**.



Now we can select what fields we want to use in the Local Storage. Because we do not always need all the fields on our local device and want to keep our Local Storage as small as

possible. In this demo we select all the fields.

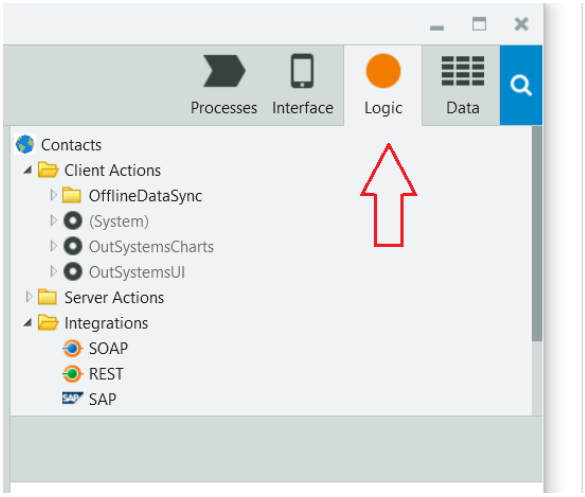


Now we are ready with the Database.

4. Synchronize the Local database

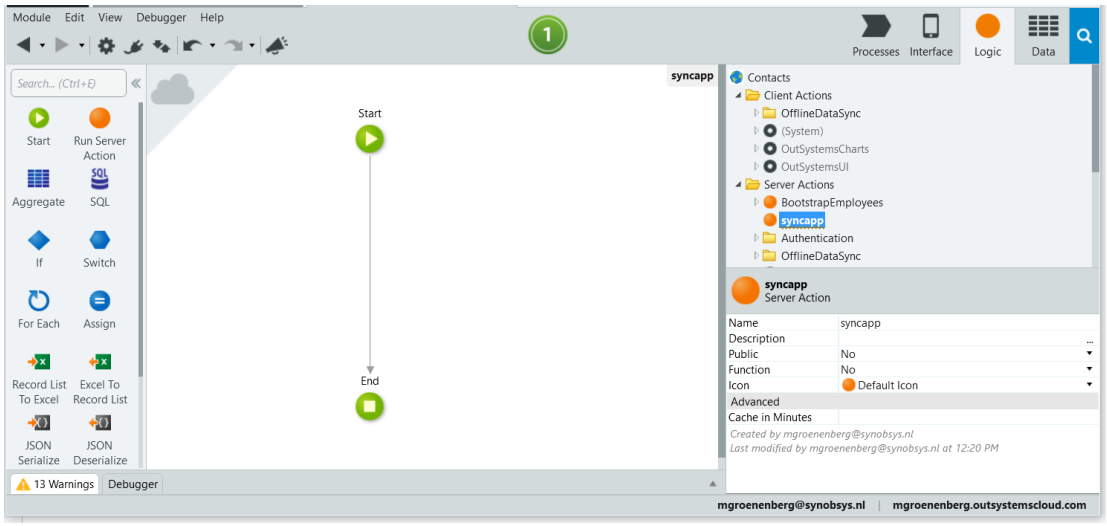
Because the **Database** is somewhere in the cloud and our **Local Database** is on our device we need to import the records from this cloud database. In this Demo we only import the records one time and do not create a real synchronization for these databases.

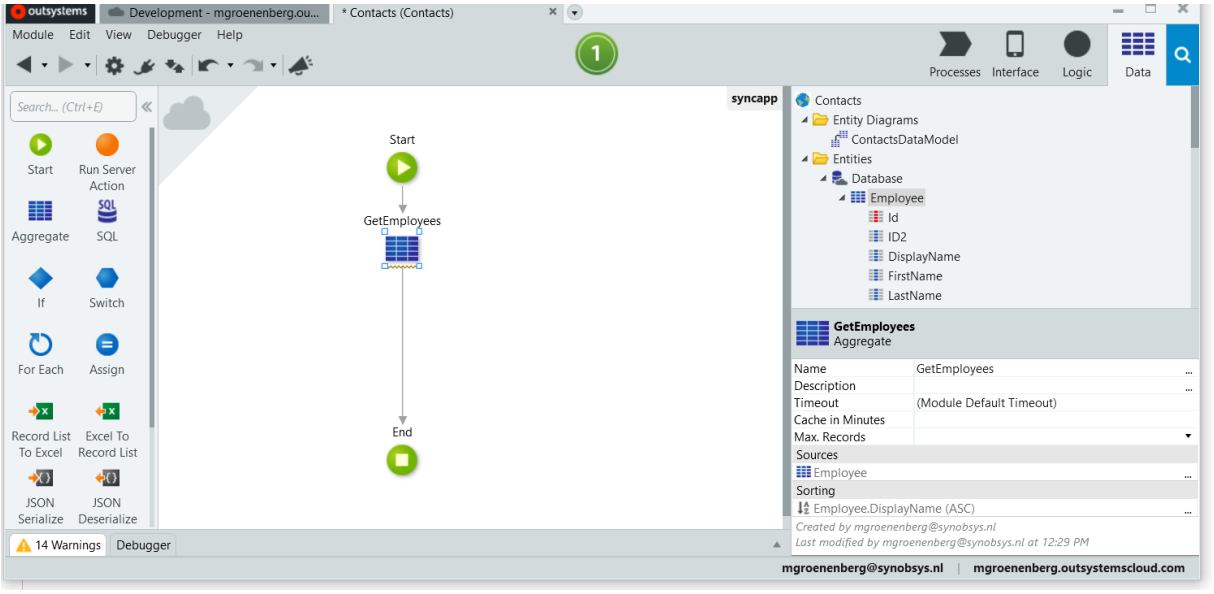
Open the **Logic** perspective



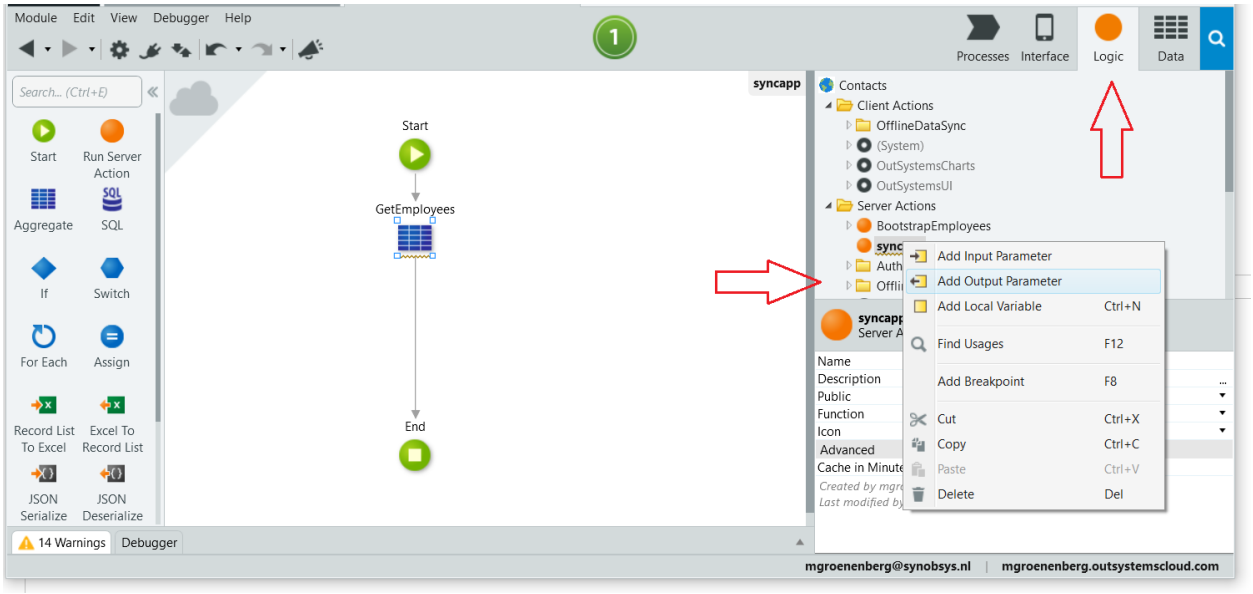
Go to **Server Actions** right click and **Add Server Action** and name it `syncapp`.

Now when you are on your new created action open the **Database Tab** and drag and drop the **Entity `Employee`** on the line of your action. An Aggregate **`GetEmployees`** is created which will retrieve all the records from `Employee`.

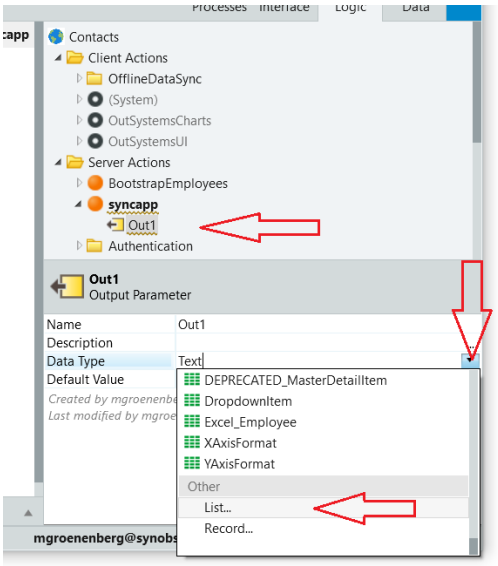


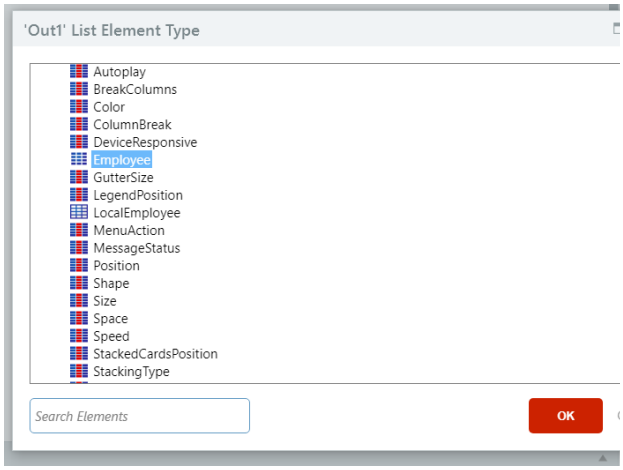


Go back to the Logic tab. Select your action `syncapp`, right click on this action and **Add Output Parameter**. Let the name be "`Out1`".



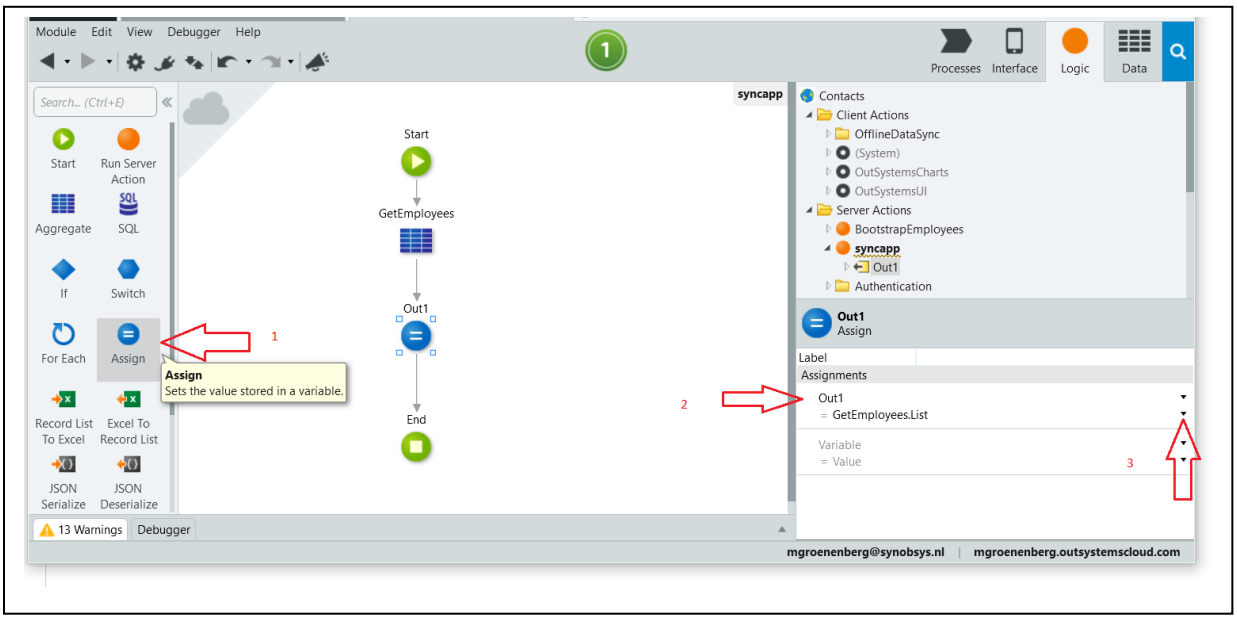
Change the **Data Type** of this Output Parameter "`Out1`" to **List** and after that select the Entity `Employee`.





In your action `syncapp` select from the left pain the **Assign**, drag and drop this on the action line under the **Aggregate**.

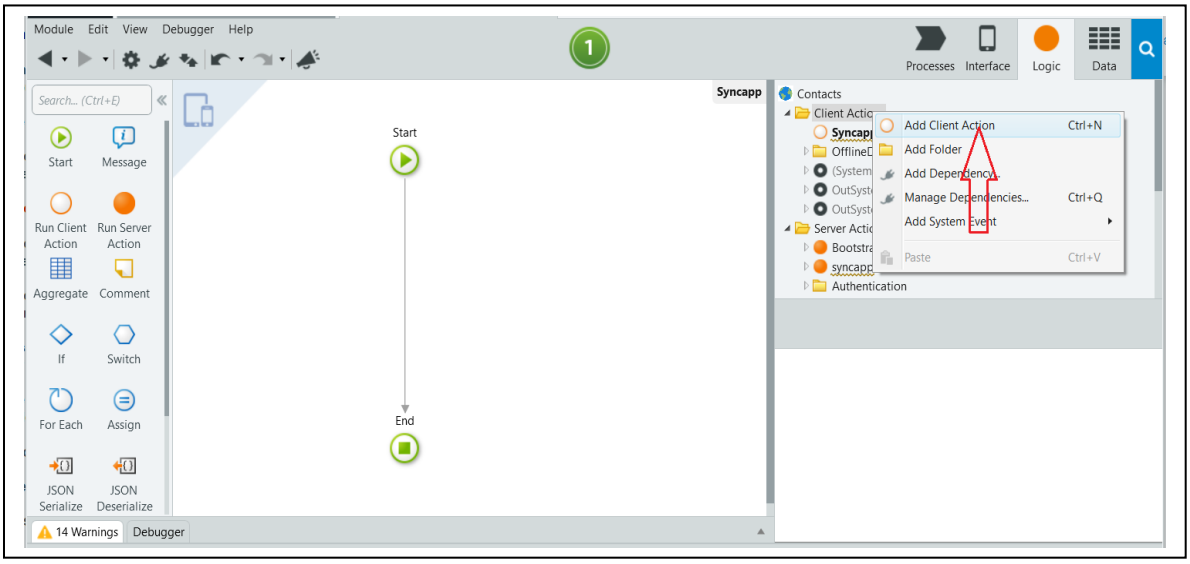
In the Assignments pane (in the right corner) select as variable the `Out1` parameter and as input the list from `GetEmployees`.



Your `syncapp` Server side is ready. Now on the Client side we also need an Action.

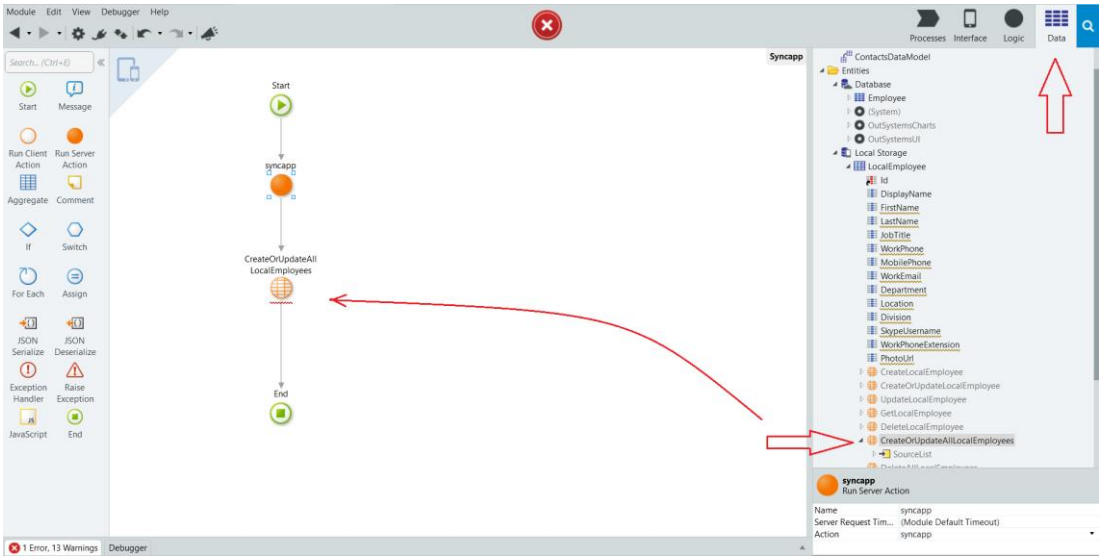
5. Syncapp Client Side.

Right click on the **Client Actions** and **Add a new Client Action**. Name it also `syncapp`.

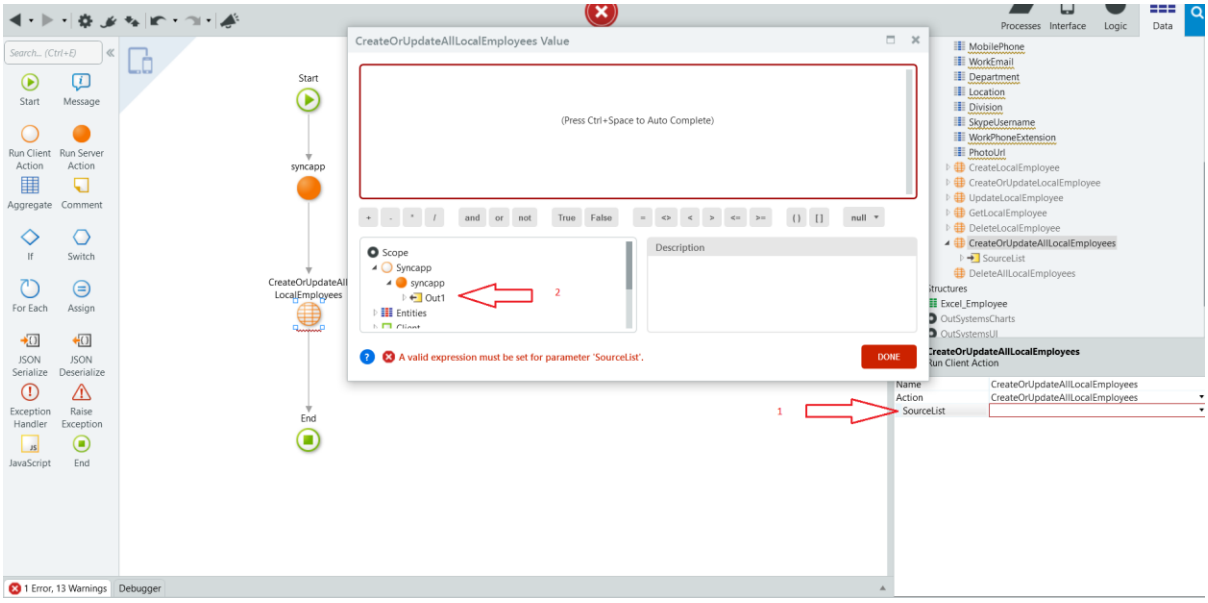


Drag and drop the early created Server Action `syncapp` on the line of the action diagram.

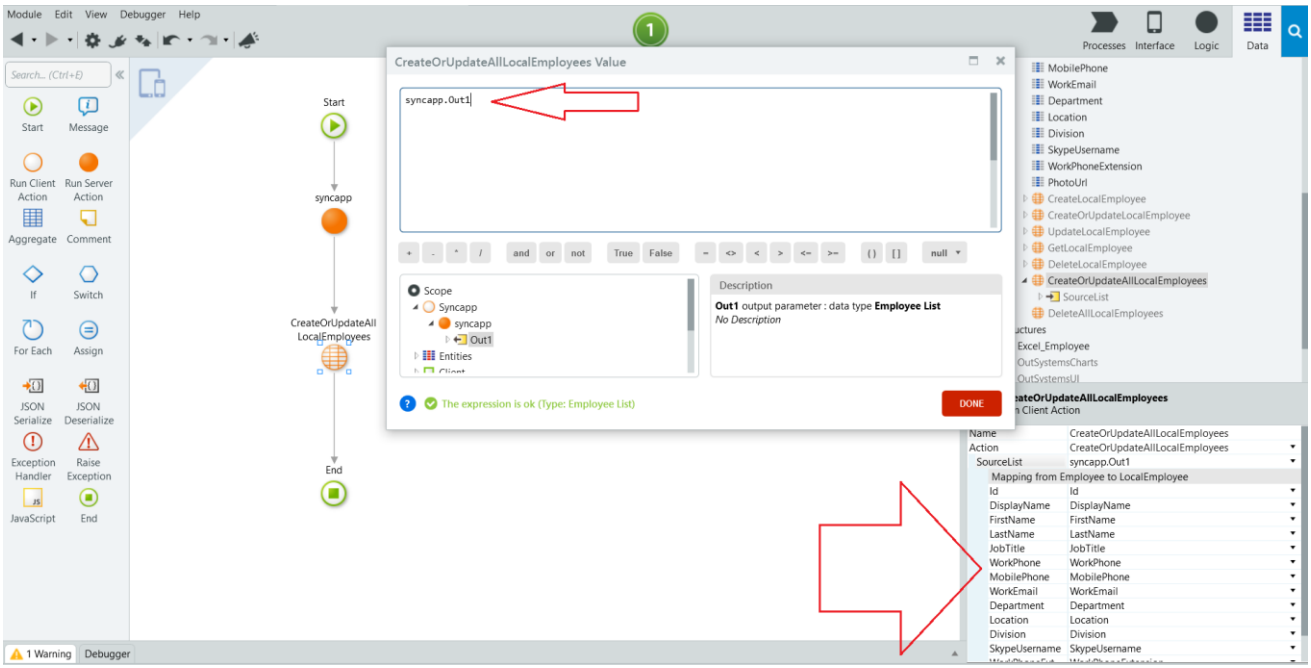
Now open the **Database tab** and go to the **Local Storage**, Select the Entity `LocalEmployee` and go to the action `CreateOrUpdateAllLocalEmployees`, drag and drop this action on your action diagram. (Aware: there is also a `CreateOrUpdateLocalEmployee` for fetching a single record. Ignore this one!)



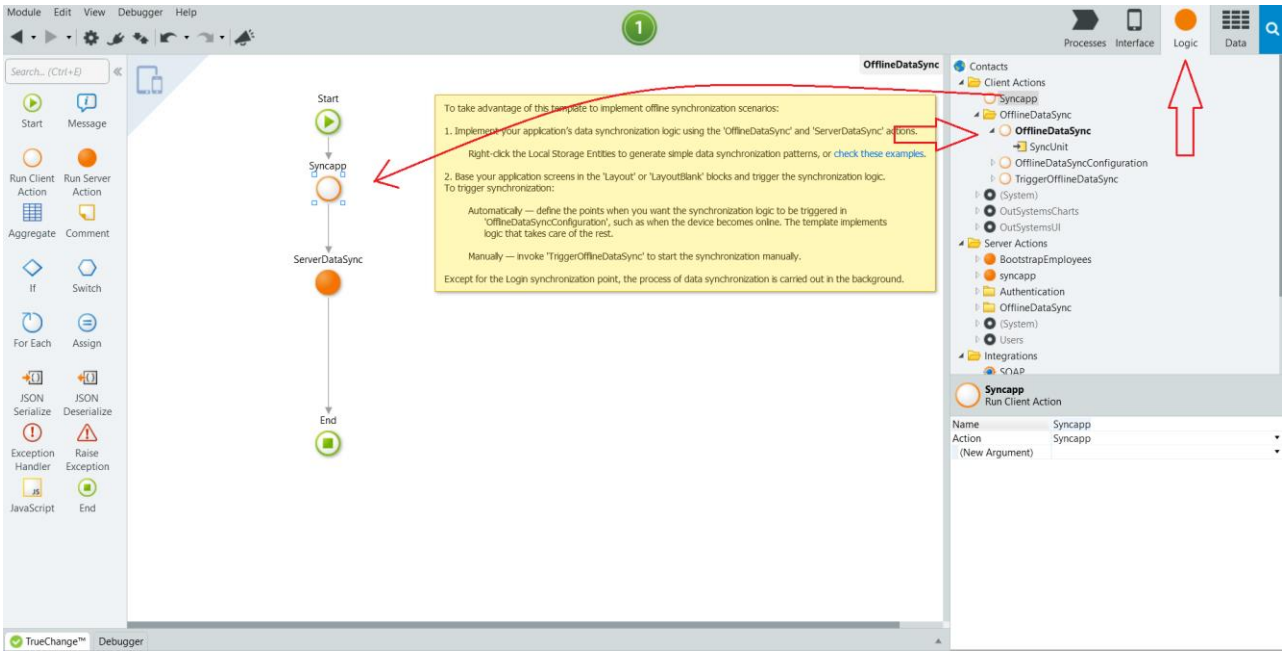
Double click on the **Source List** of this action so the **Expression Editor** of OutSystems pop's up. Select the output parameter `Out1` of the server action.



Note that all the fields from the server action are mapped to the fields of the client action.



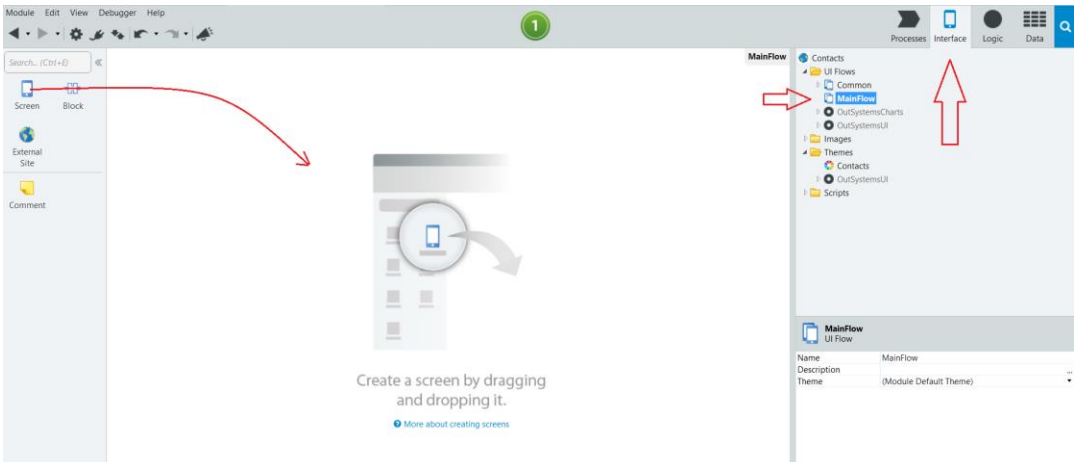
On the **Logic tab** open the action `OfflineDataSync`.
Now drag and drop your new `syncapp` to this action.



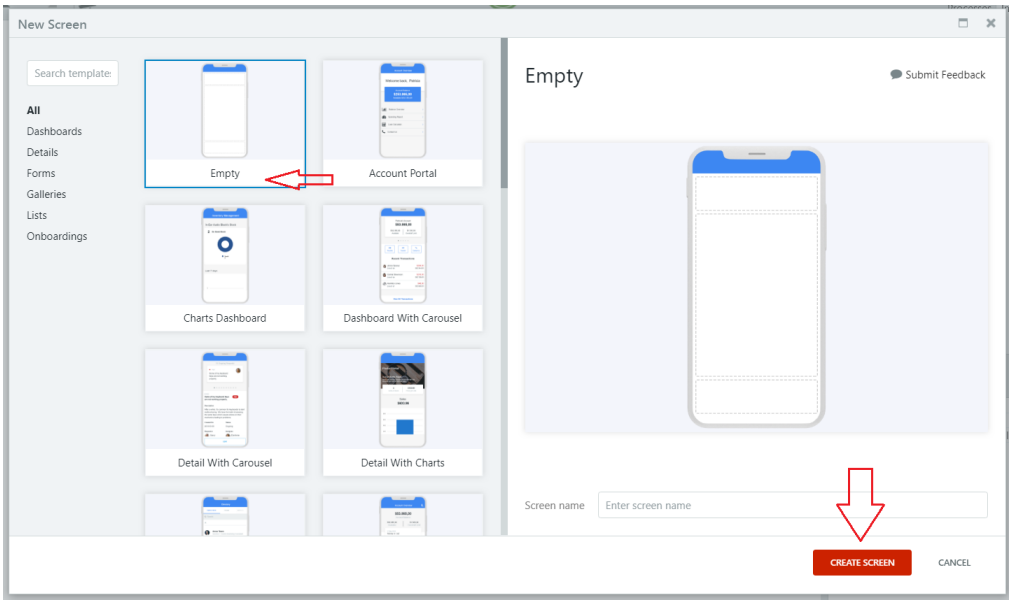
That's all for the database and synchronization now lets' go to the Screens.

6. Screens

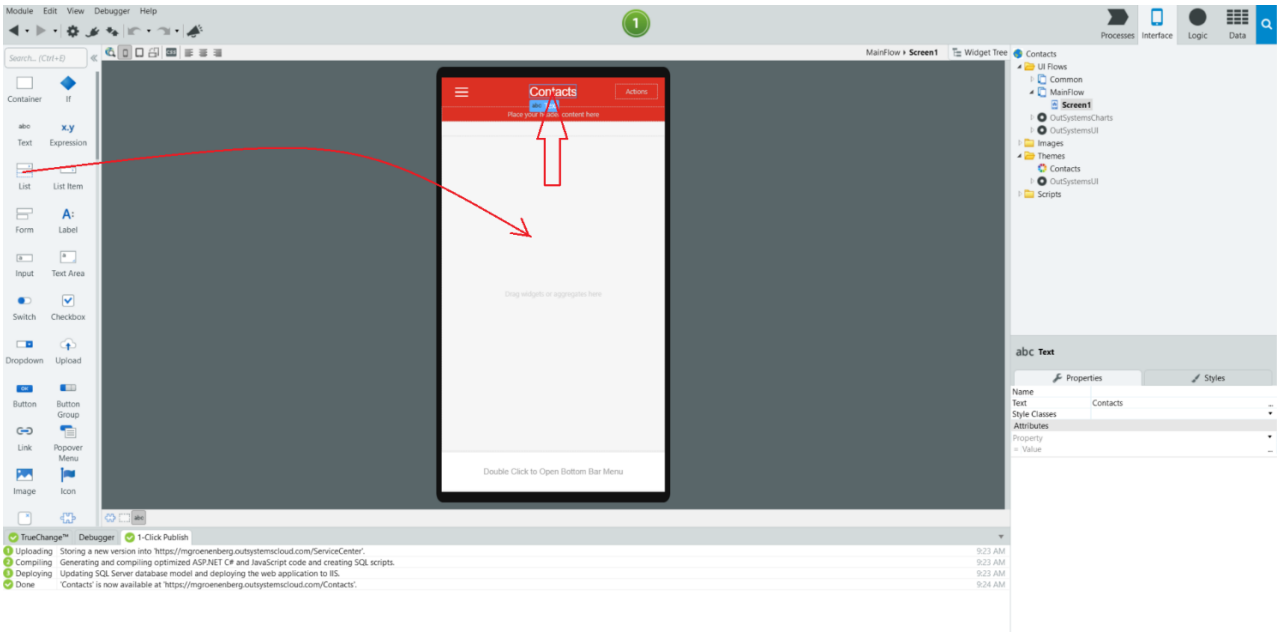
Go to the Tab **Interface** (right top corner) and select and double click the **Mainflow**. From the Left pain select the **Screen** Icon and drag and drop it on your screen.



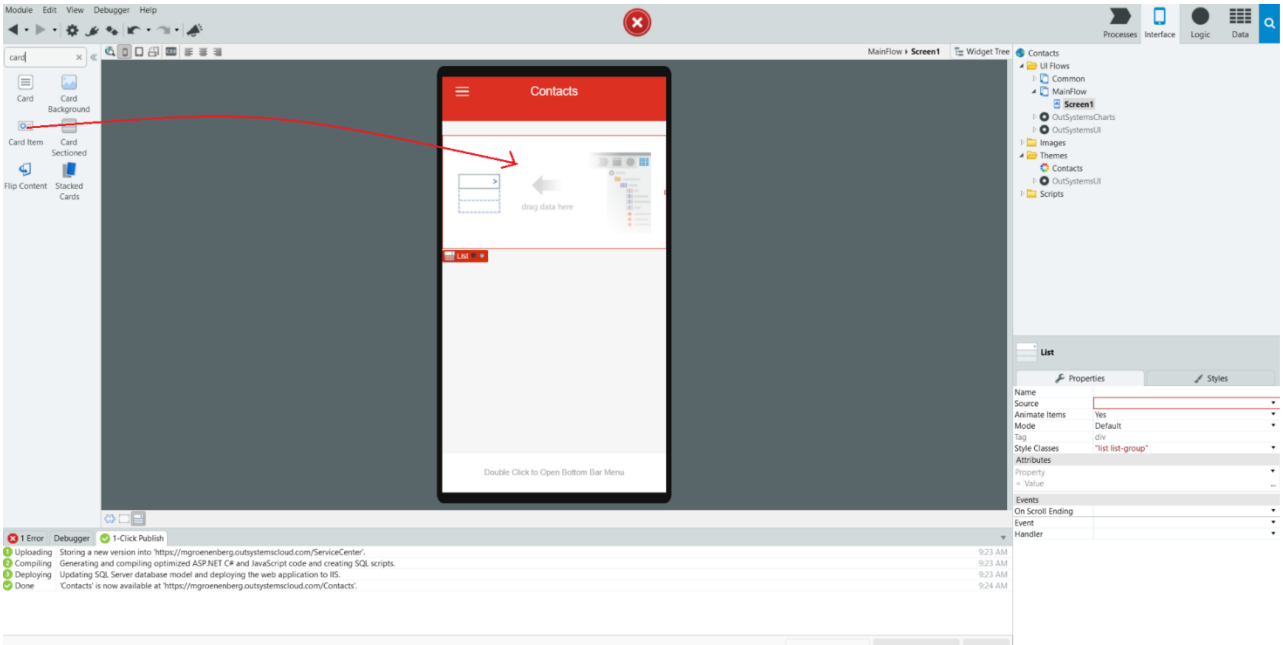
Select the empty screen and hit the Button **CREATE SCREEN.** (You are allowed to enter a screen name. If you don't, Outsystems will apply name "Screen1" automatically).



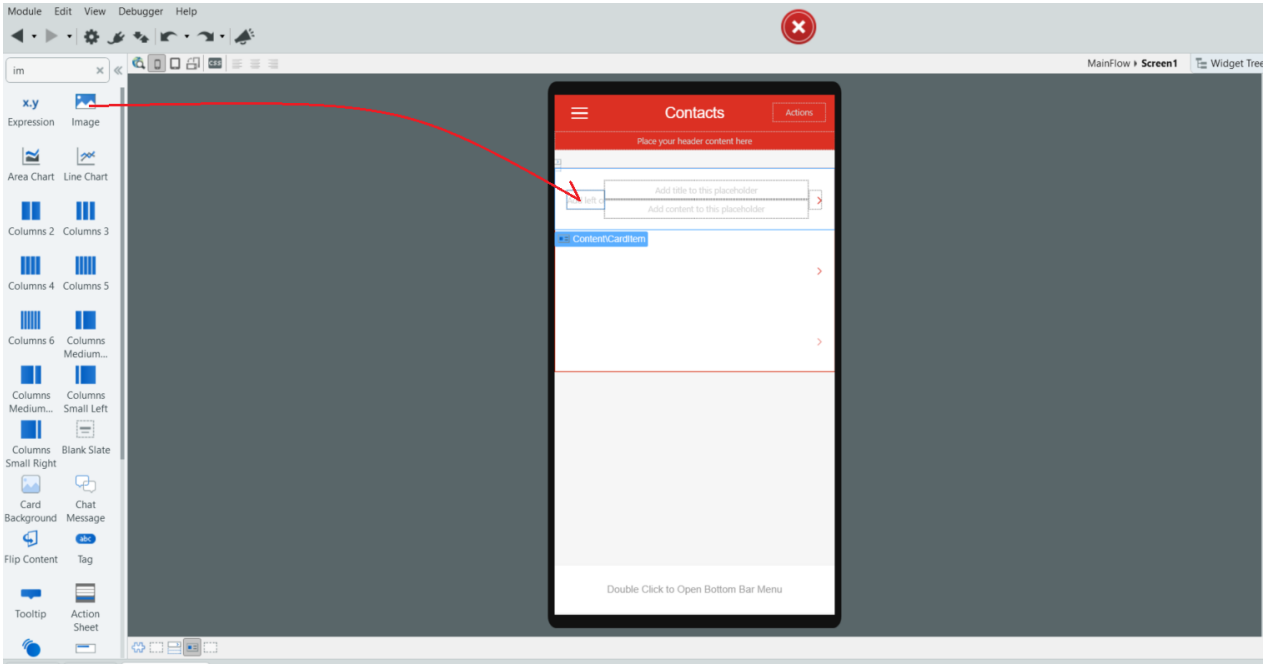
The screen design panel is now activated. At the very top you find a placeholder called “Title”. Change it into `Contacts`. Then drag and drop from the left pane the widget **List** on the main section of your screen.



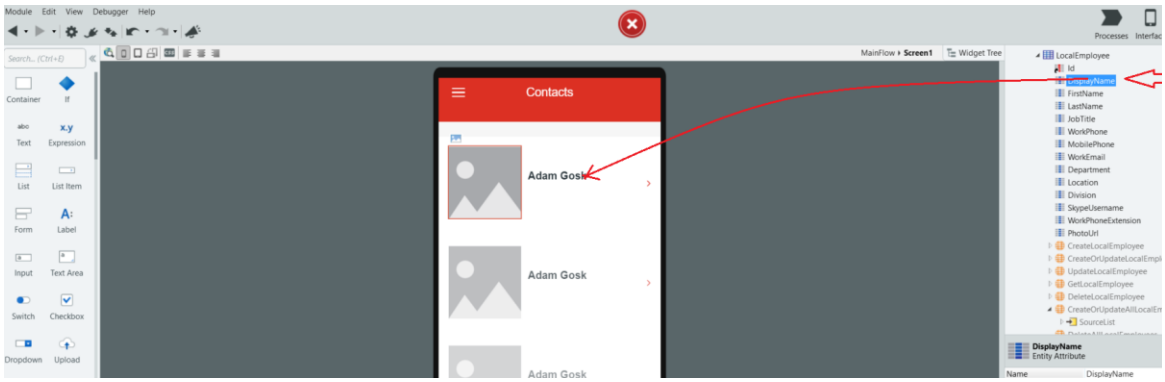
Drag and drop from the left pane the widget **Card Item** on the **List** you just created.



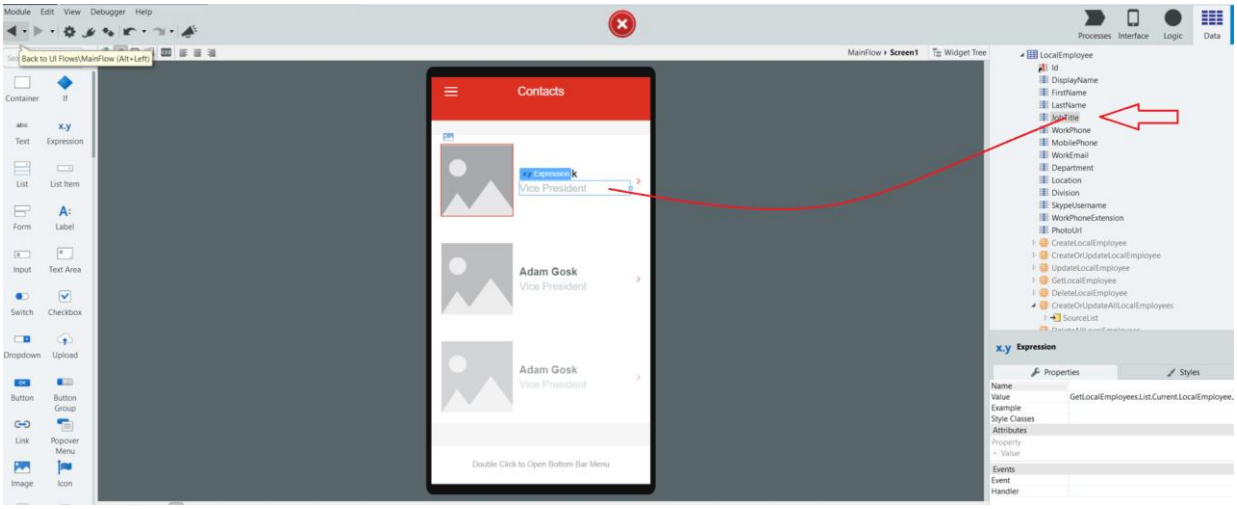
Drag and drop from the left pane the widget **Image** on the left of the **Card Item** you just created.



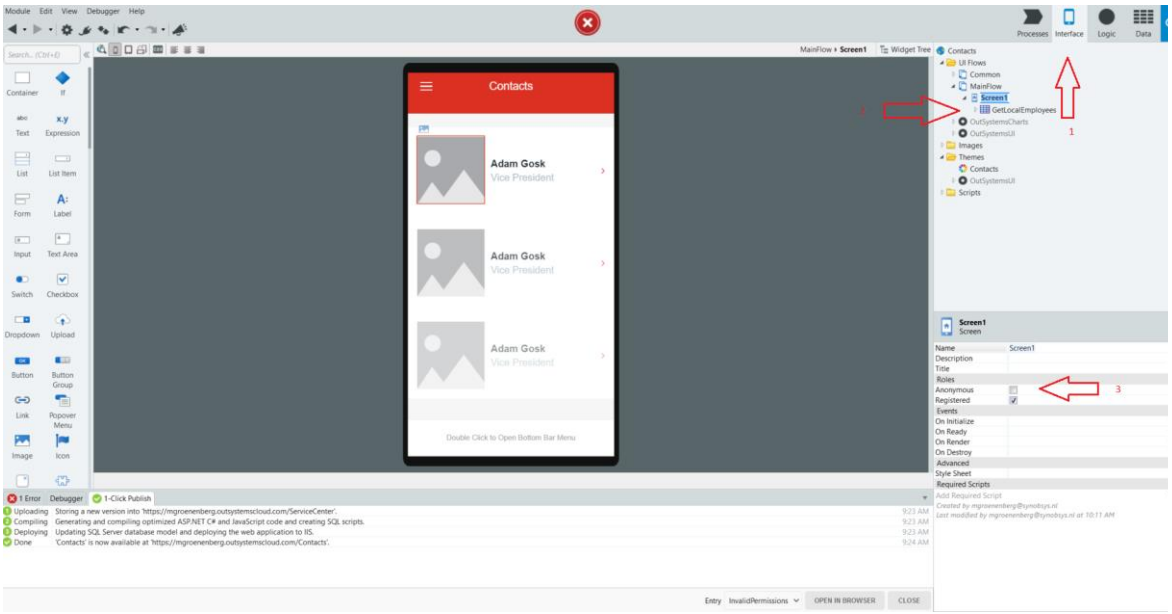
Now open the **Database tab,** go to the **Local Storage,** select file `LocalEmployee`, click it open to see all attributes. From this file select and drag and drop the field `DisplayName` on the first field of your list.



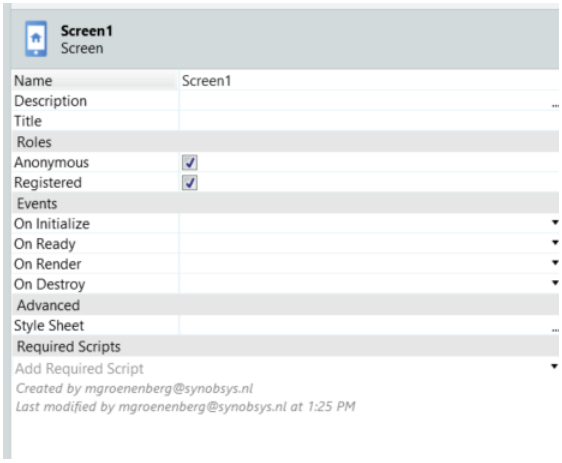
Drag and drop the field `JobTitle` to the second field of the list.



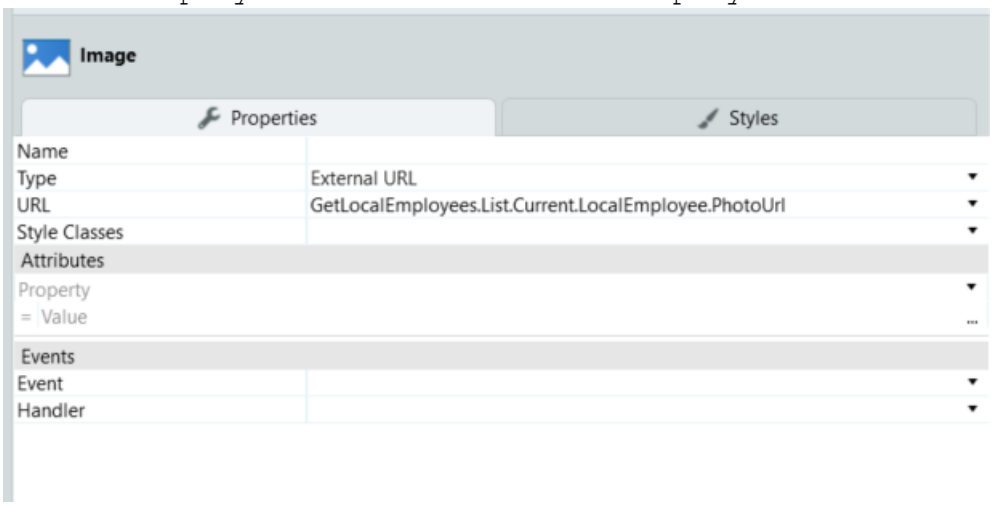
Go back to the **Interface Tab** and notice that under your `Screen1` a so-called aggregate is automatically created by OutSystems named `GetLocalEmployees`. This is a data-fetch we are going to use in the next steps.

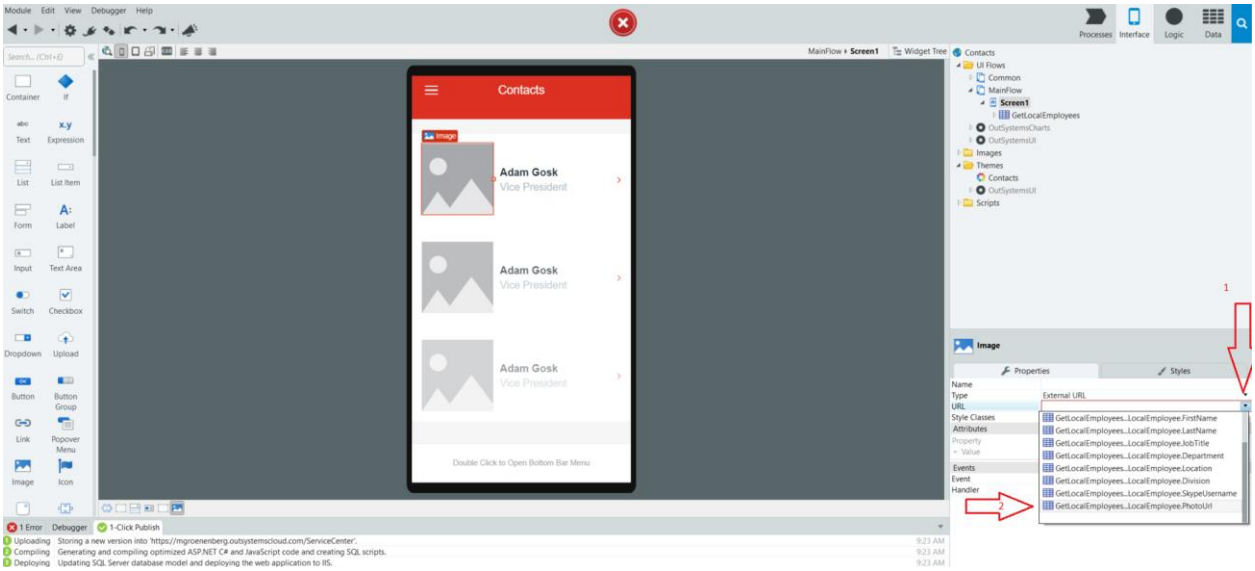


Outsystems is creating Roles automatically. Because we do not want to consider about authorization for this screen click the select box for Anonymous. See next page.

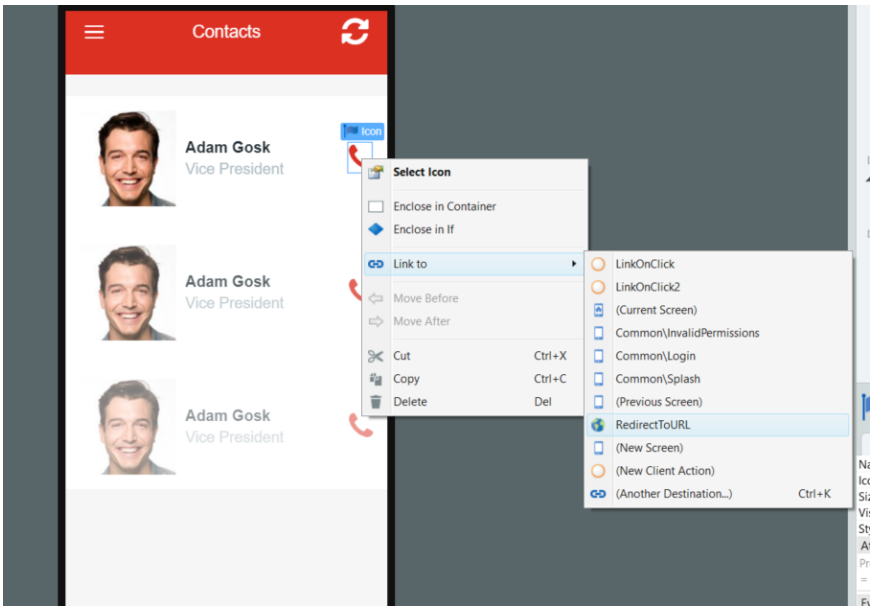


Now click on the **Image** in your **List,** go to the properties pane at the right and in the Type property, select type **External URL** and for the **URL** itself select field `GetLocalEmployees.List.Current.LocalEmployee.PhotoUrl`.



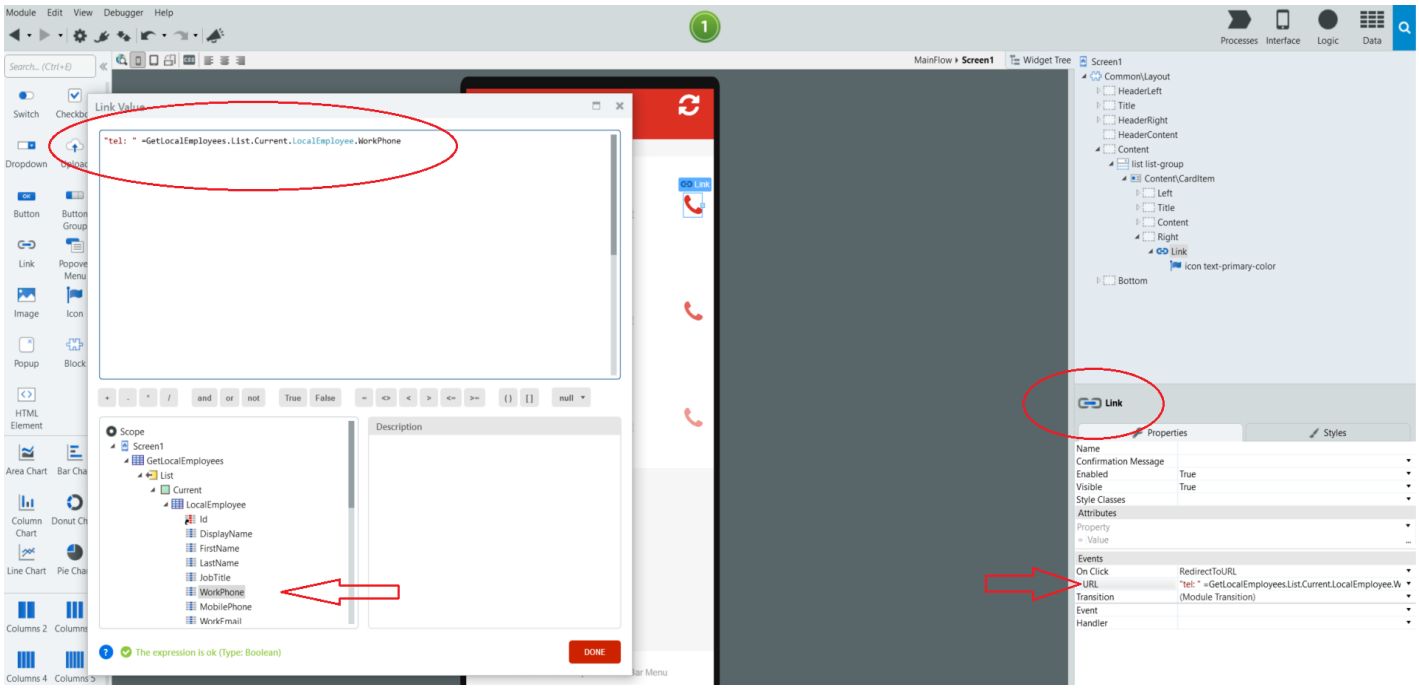


Now click on the Icon “>” in your List, go to the Properties pane at the right and change the Icon “phone”. Set the Size as “2x font size”. Right click on the icon in the center pane and select **Link to > RedirectToURL**

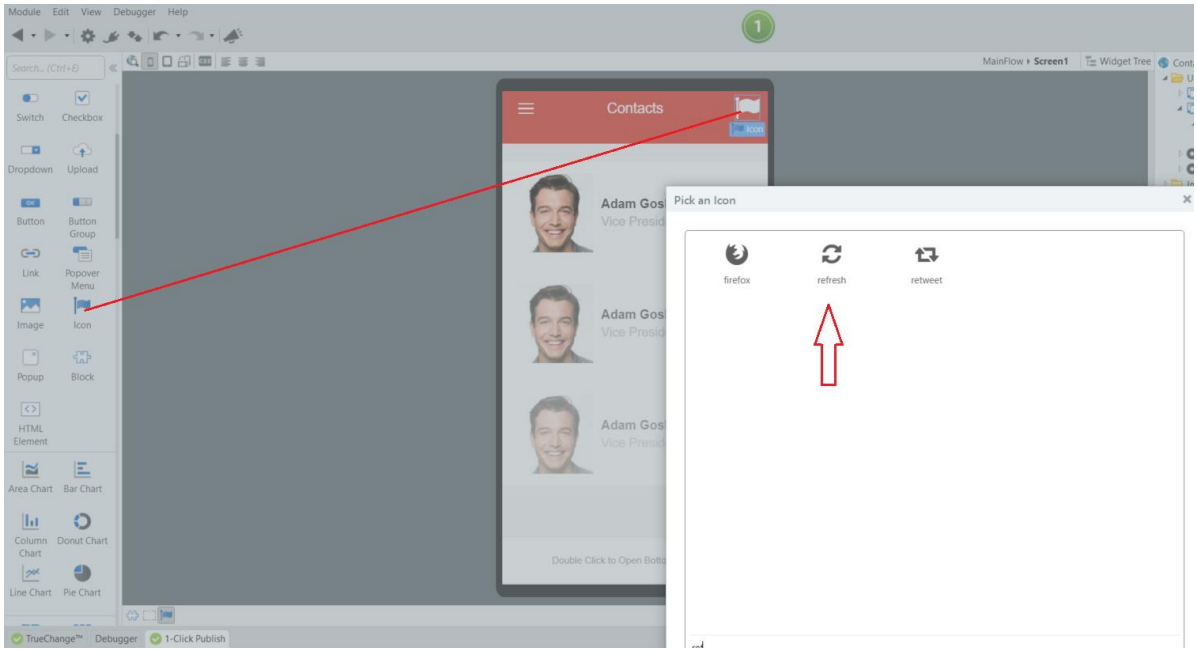


On the **Properties** of this link **Event** double click on the **URL** parameter and add the text: “tel: ” + <select the workphone from the current LocalEmployee> so it looks like this:

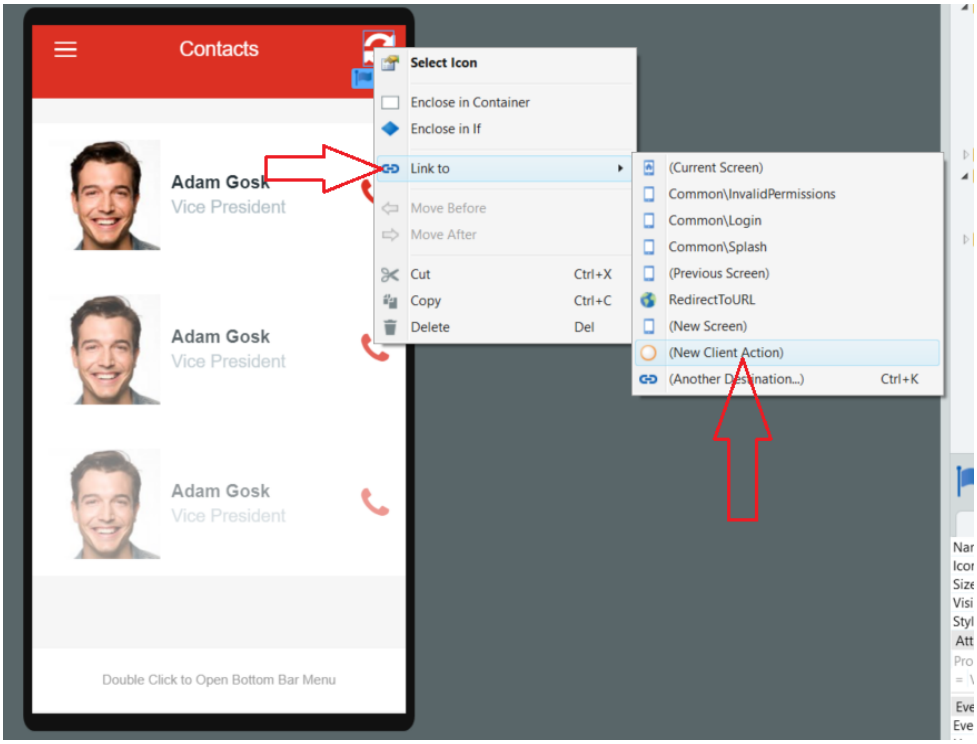
```
"tel: "+GetLocalEmployees.List.Current.LocalEmployee.WorkPhone
```



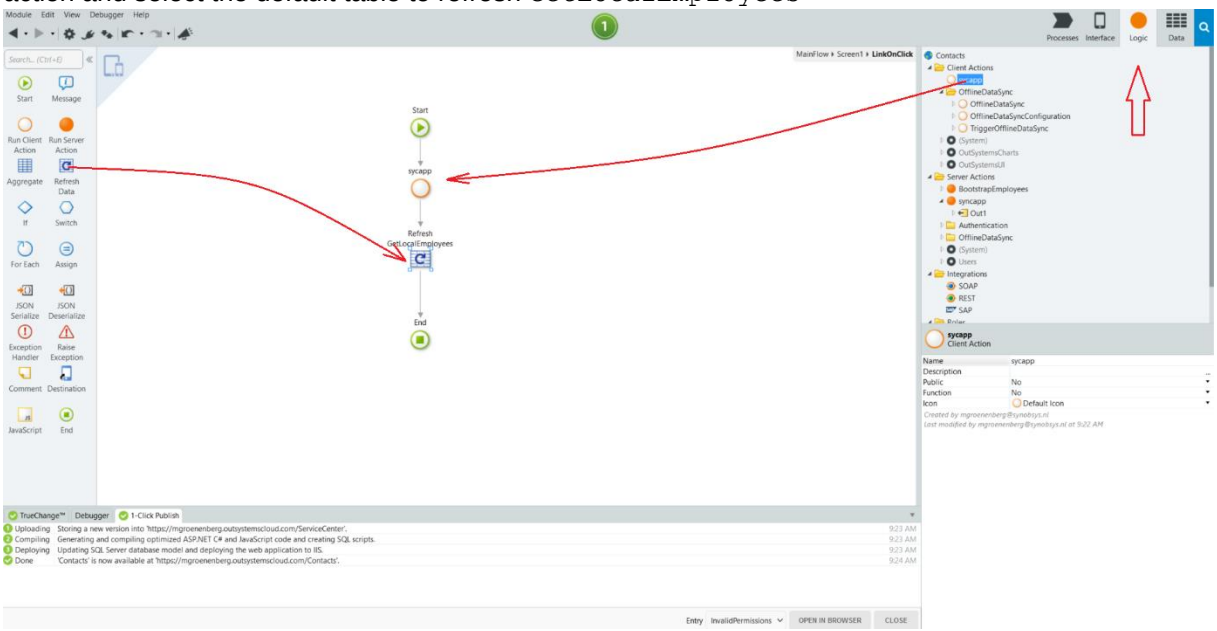
Drag an **Icon** from the left pain to the Action part of the Header and select the **Refresh** Icon.



Right click on the new **Icon** and select **Link to** and then **New Client Action.** A new action is created on your screen named `LinkonClick`**.** Open this newly created Action from within the Interface tab. Then go to the **Logic Tab** and drag and drop the **Client Action** `syncapp` to the action line of this action.



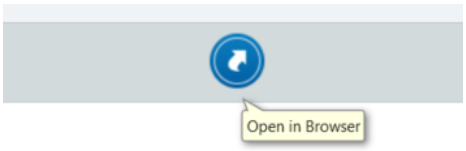
After the `syncapp` action we insert a **Refresh Table** by dragging and dropping it under the `syncapp` action and select the default table to refresh `GetLocalEmployees`



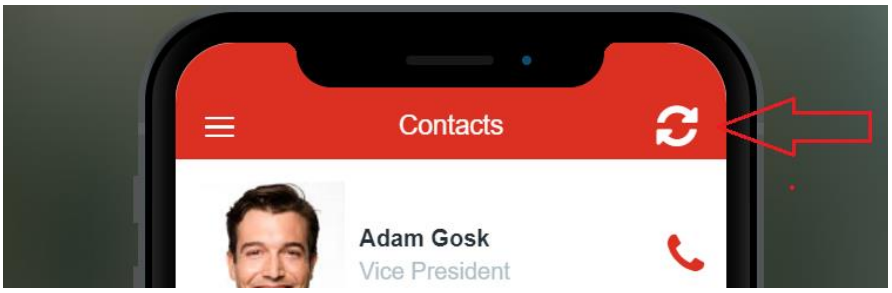
Now we have created a button that will sync the data from the cloud database to our local database and refreshes our screen.

You can test your screen now by pushing the 1 click publish button.

After the image becomes blue you can open it in a browser to test.



When it opens in the browser you can push your refresh button in the header to get your data on the screen.



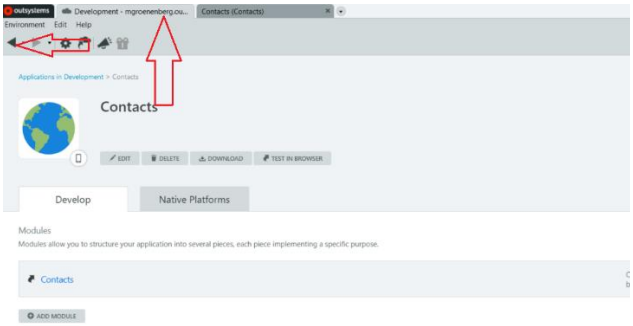
7. Integrating Plugin from the Forge.

You can consume any Plugin which is available in the Forge (an Outsystems environment in where one is allowed to download apps build by others).

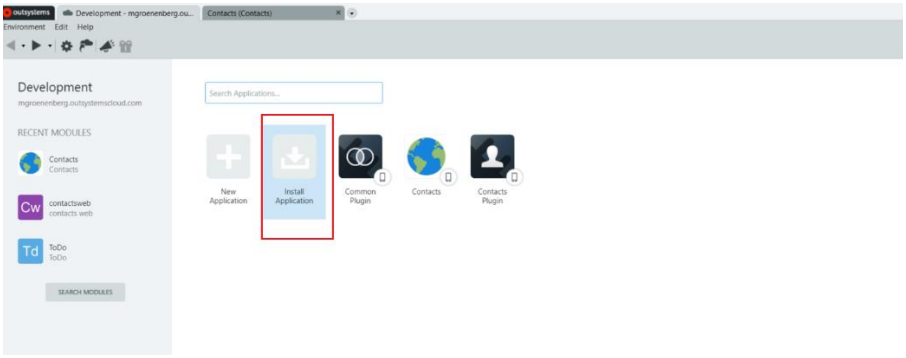
As an example we want to add the contact from our employee list to the contact list of our own mobile phone when we press on the picture (after done so, you can go to your Contacts on your phone to delete the contact again).

To do that we need 2 plugins to be installed.

Go back to your personal development environment by clicking the tab at the very top. Then click on the “Applications in Development” link (left above). You can also use the back button.



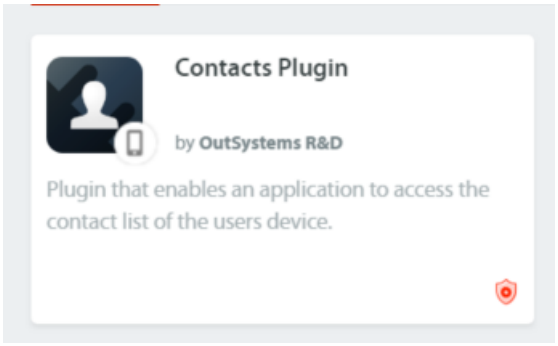
Now you should be back in the development home screen from where you started.



Click **Install Application** and browse the Forge for the **common plugin**.

Select and Install this plugin.

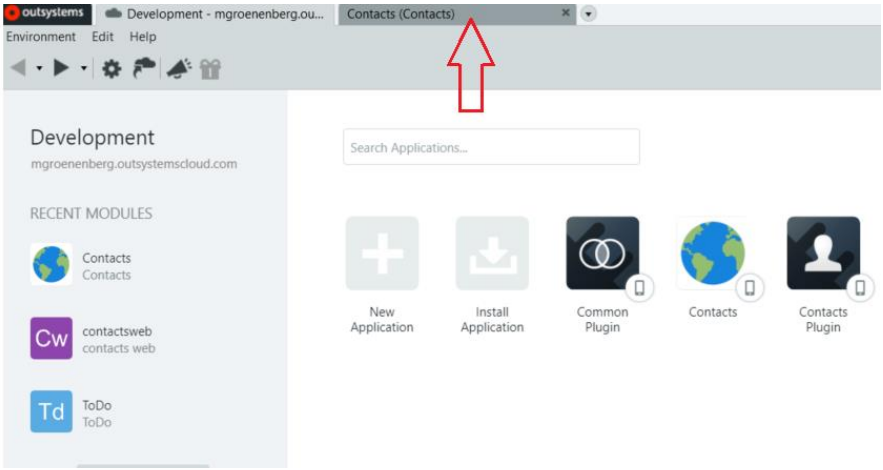
Next search for the Contact plugin.



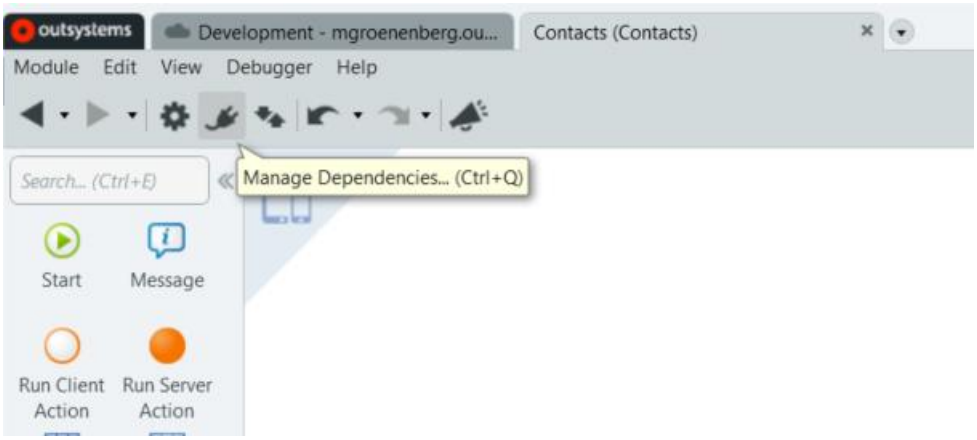
Select and Install this plugin.

Check on your **personal development** home page that the plugins are installed.

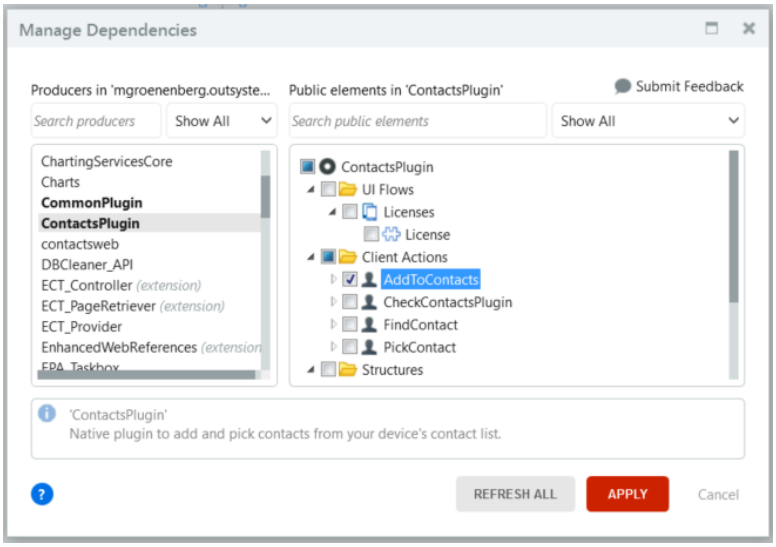
Go back to your **Contacts** Application by selecting the tab on top of your screen.



In your C**ontacts** application click the button **Manage Dependencies.**



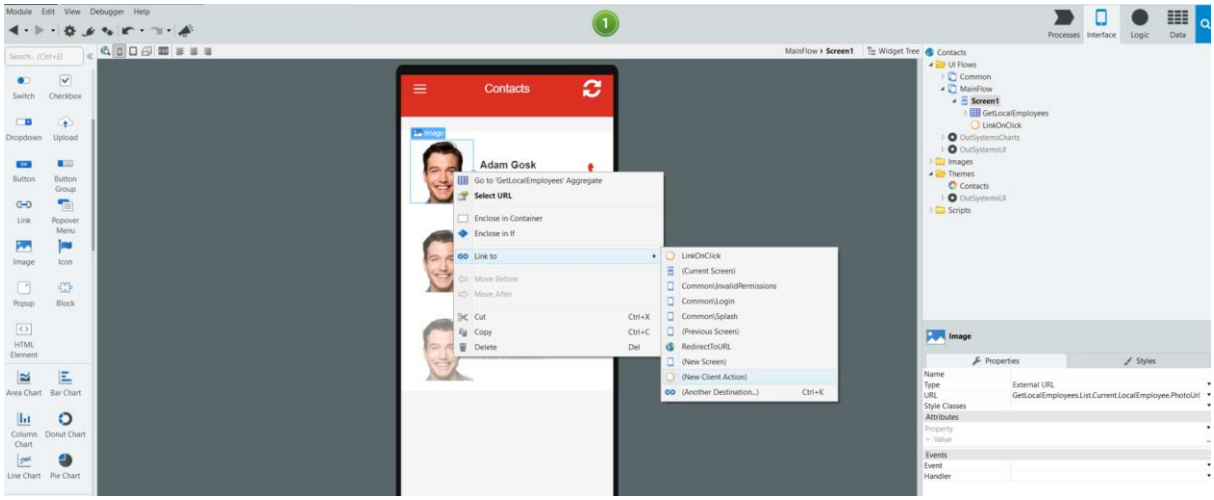
In the popup search for the **ContactsPlugin** and select **AddToContacts**.



APPLY the changes.

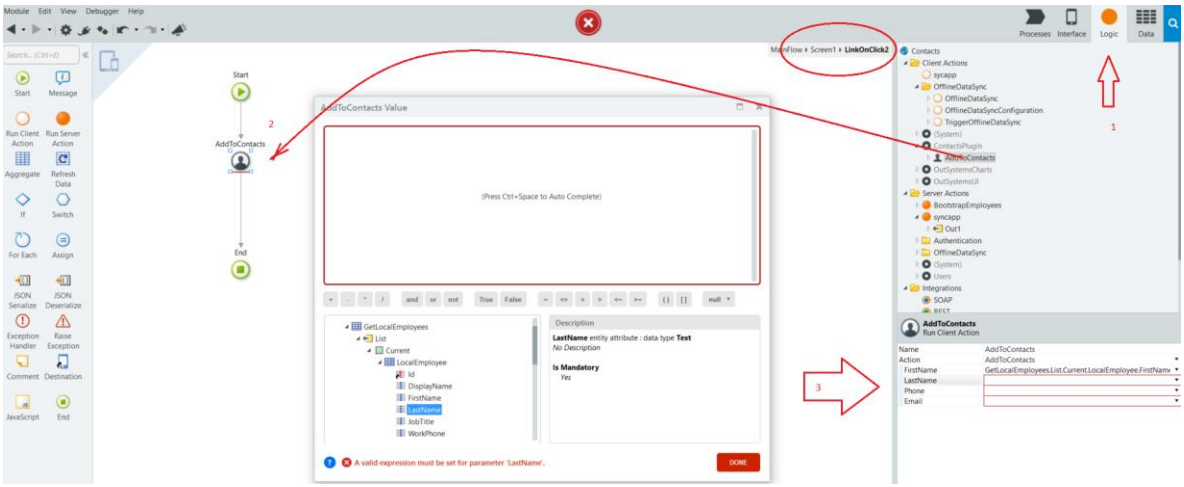
Go back to your `screen1` and right click on the picture in your list.

Select **Link to** and then **New Client Action.** A new client action `LinkOnClick2` is created.

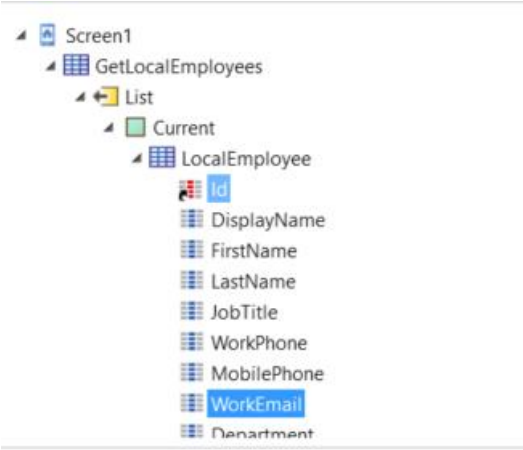


Open this action by double clicking it.

Open the **Logic tab** and drag and drop the action `AddToContacts` on the action diagram.

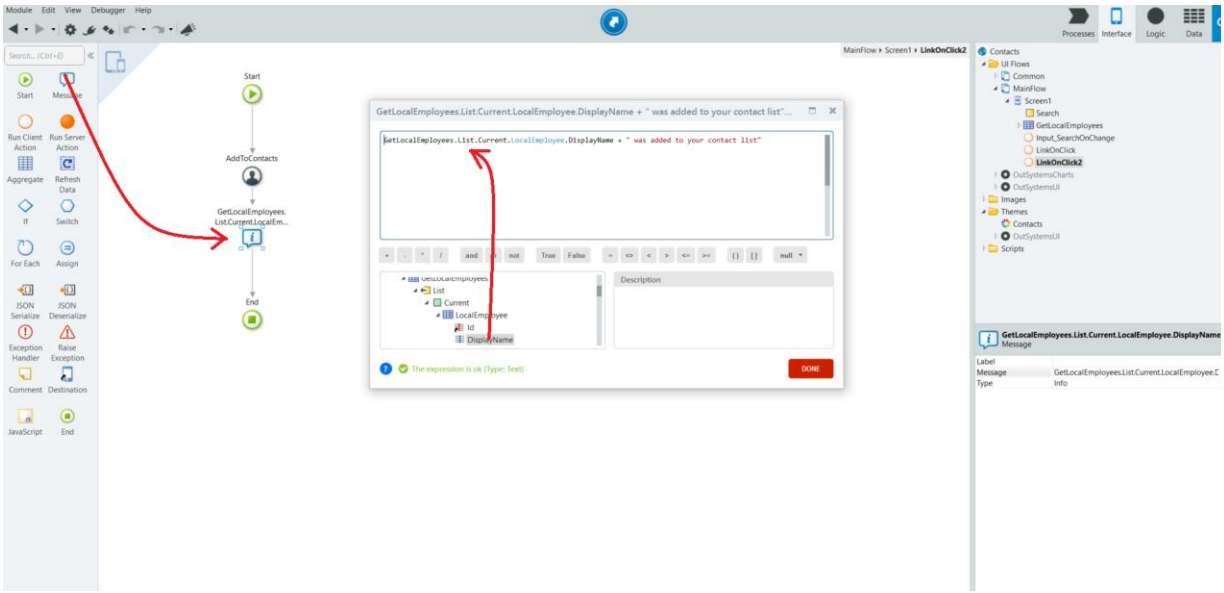


Add the 4 parameters that this action needs by using the **expression editor** (double click on each parameter and then select the value from within the popup).

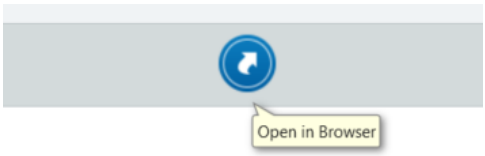


As shown below: add a M**essage** from the left pane to the action to inform the user. Edit the M**essage** text: First select from the list the current `DisplayName` and add a text like:

GetLocalEmployees.List.Current.LocalEmployee.DisplayName + " was added to your contact list"



Now you are ready to test your application by publishing and after that start the browser.



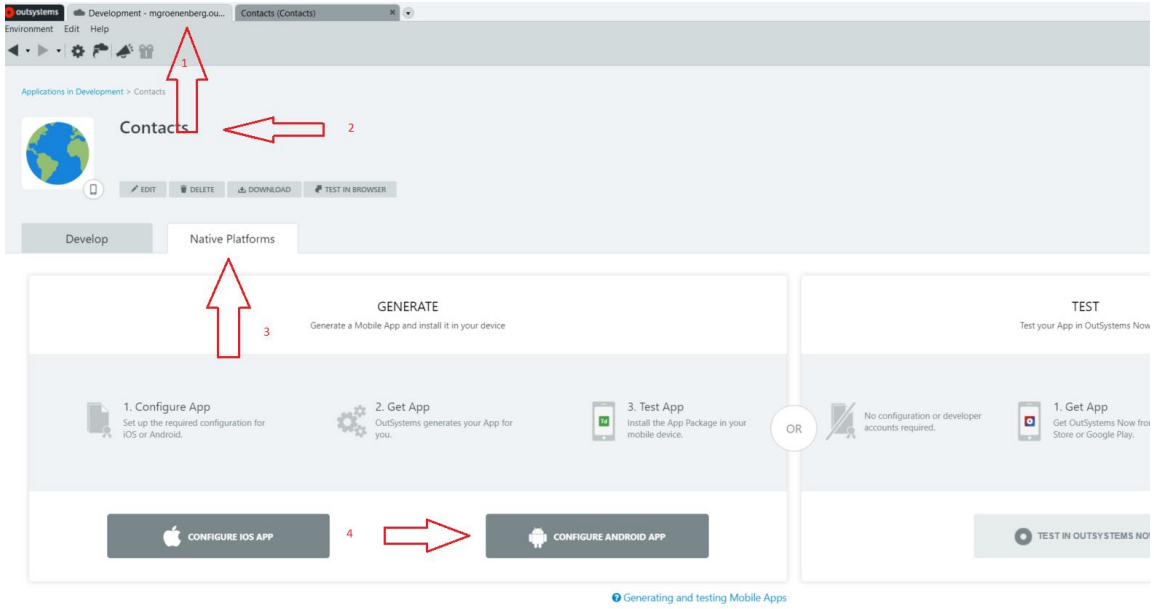
To prove it works we have to bring it to your mobile device. See next….

8. Run your application on your Mobile Device (Android)

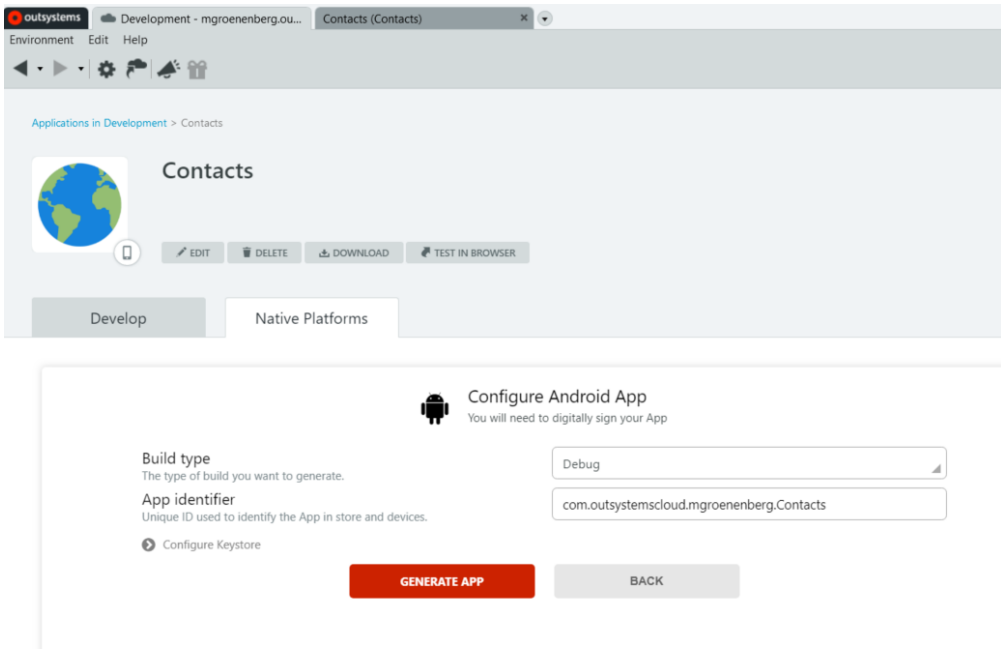
Open the **Development** home page by selecting this tab in top of your screen.

Select the **Contacts** application and the **Native Platform**.

Select **CONFIGURE ANDROID APP**



Select **GENERATE APP**:

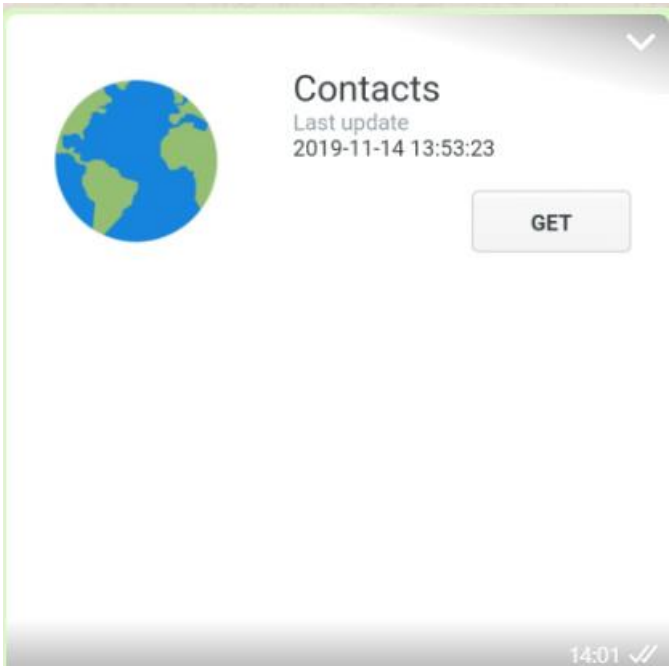


OutSytems will now generate the Android App.

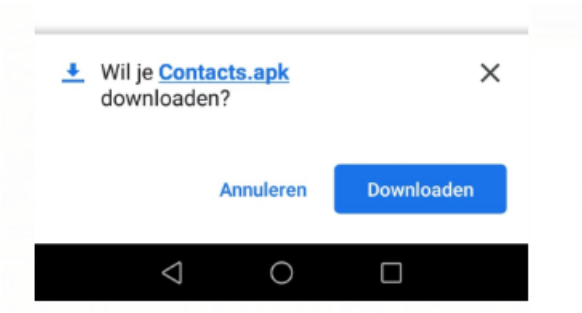
With a QR-code reader application (https://play.google.com/store/apps/details?id=net.grbot&hl=nl) you can with your mobile device directly open the website where your application is stored. Otherwise copy the created link (as presented to you by Outsystems) and go there with your browser on your mobile device.

https://<yourname>.outsystemscloud.com/NativeAppBuilder/App?AppKey=49e13ff6-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Open the website and GET the application



Download the application and Install the application.



Accept all messages you get from your device.

Your application is now ready to test!

9. Run your application on your Mobile Device (iOS)

1) Before You Start

To generate your iOS mobile app you must be enrolled as an Apple Developer. If you haven't enrolled yet, learn how to enroll as an Apple developer.

To test your app, you must have a certificate generated and configured in your Apple Developer account. The exact type of certificate depends on the developer program you enrolled in:
Select:
— App Store and Ad Hoc certificate for the Apple Developer Program
If you don't have one, learn how you can create a certificate.

2) Create a Certificate

Log in your Apple Developer account and select the option **Certificates, IDs & Profiles**.

The certificates you can use:

- For a development certificate, use the iOS App Development certificate type.

To create the certificate (.p12 extension format):
Generate the Certificate Signing Request (CSR) File

To create your certificate you need to provide a CSR file. Apple will provide you with instructions on how to create one if you have an Apple machine, but if you don't have a Mac here's how you can do it on a Windows machine:

1. Download the most recent version of OpenSSL.
2. Start a new command prompt (Start > Run > cmd.exe).
3. Execute the command to set the folder used on the OpenSSL installation:

```
4. set OPENSSL_CONF=<path_to_openssl_bin>\openssl.cfg
```

   where <path_to_openssl_bin> can be C:\OpenSSL-Win32\bin or C:\OpenSSL-Win64\bin.
5. Now call the OpenSSL to create the CSR and the private key:

```
6. <path_to_openssl_bin>\openssl.exe req -out
   <CSR_file_name>.csr -new -newkey rsa:2048 -nodes -keyout
   <privateKey_name>.key
```

where <CSR_file_name> is the name for the CSR file and <privateKey_name> is the name for the private key file.

After you run the command on your computer, it will generate a new CSR file with a public key embedded and a private key.

3) Download the CER and Create the P12 Certificate

Now that you have created the CSR file, go back to the Apple Developer Portal and upload the file. Follow the next steps on Apple's page to create a new Certificate (CER file format). This certificate includes both public and private keys and can be used for multiple apps.

Be sure to back-up the CSR and the CER files in a safe place.

If you're using a Mac, download the CER file, install it and export to the P12 file format. If you are using a Windows machine, open the command prompt and execute the following commands:

```
<path_to_openssl_bin>\openssl.exe x509 -in <certificate_cer>.cer -
inform DER -out <app_pem_file_name>.pem -outform PEM
```

where <certificate_cer> is the name of the certificate downloaded from Apple and <app_pem_file_name> is the name for the PEM file.

Now you'll need to use the file you've just generated to run the following command and create the P12 file:

```
<path_to_openssl_bin>\openssl.exe pkcs12 -export -inkey
<privateKey>.key -in <app_pem>.pem -out <app_p12>.p12
```

where <privateKey> is the file generated when creating the CSR file, <app_pem_filename> is the name of the created PEM file in the command executed before and <app_p12> is the name you want for the P12 file.

You will be asked for a password, create one and remember or store it somewhere — you'll need to insert it in OutSystems. The P12 file created in the last step is the certificate to generate the iOS app.

4) Provisioning profile

To create a provisioning profile (.mobileprovision extension format):

1. Access your Apple Developer account and select the option **Certificates, IDs & Profiles**.

2. Select **Provisioning Profiles**.

3. Select the type of provisioning profile you want to generate, as explained above.

4. Provide the information requested during the generation process. You will have to enter an App ID, which certificates to include in the profile and, for some types of provisioning profiles, which specific iOS devices will be allowed to launch mobile apps associated with the provisioning profile being generated. Note that if you later register additional devices in your Developer Account or add any app services, you will need to create a new provisioning profile and generate the mobile app again to propagate your changes.

5. At the end of the generation process, you will be able to download the generated provisioning profile for later use.

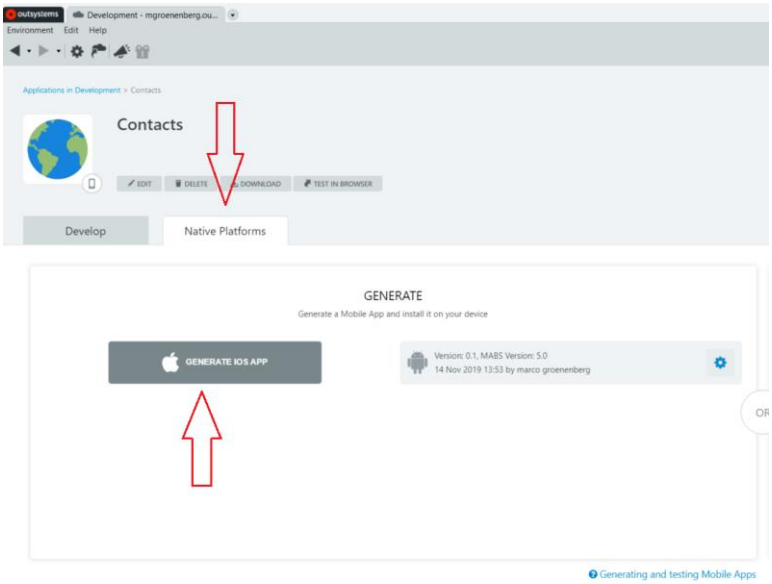
In order to generate iOS mobile apps in OutSystems you will be asked to provide a provisioning profile along with one of the certificates that you associated with it in the steps described above.

5) Generating your App

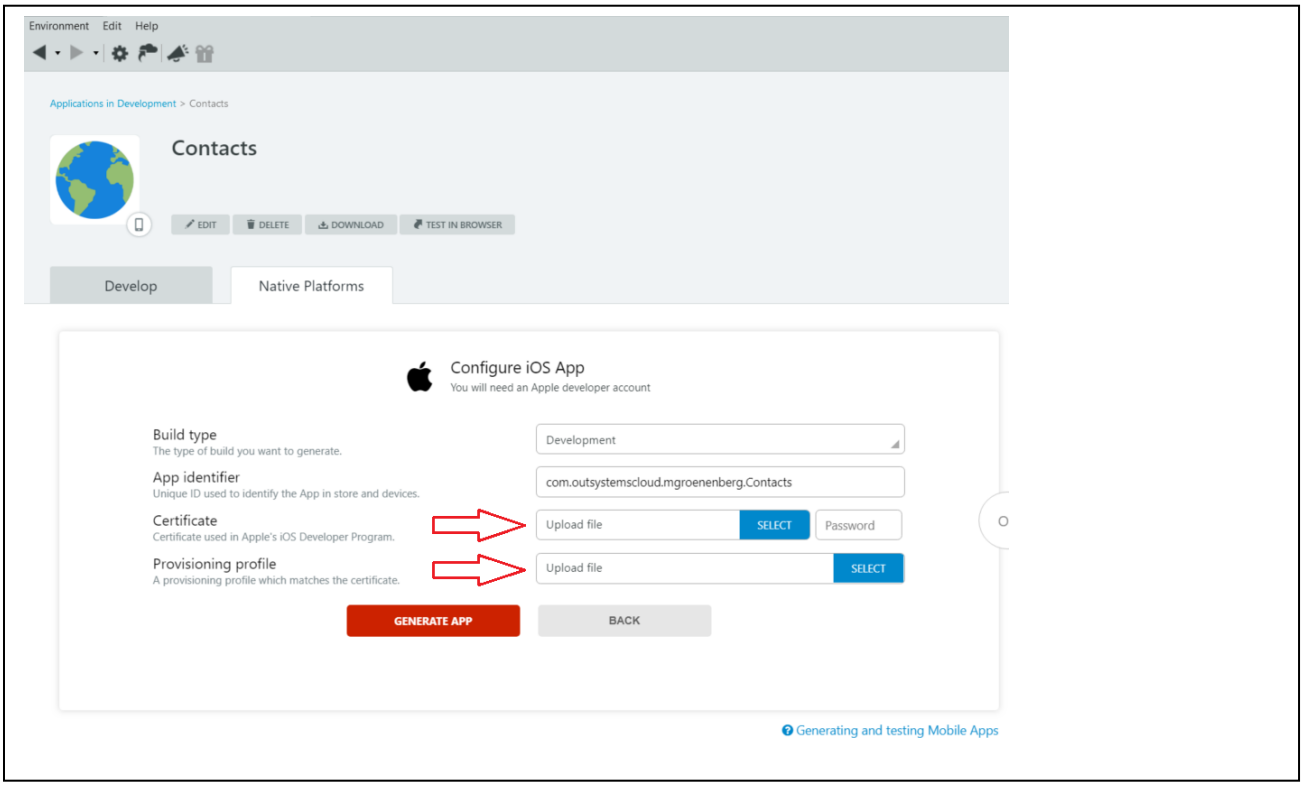
Open the **Development** home page by selecting this tab in top of your screen.

Select the **Contacts** application and the **Native Platform**.

Select **CONFIGURE IOS APP**



Now fill in the Certificate and the Provisioning profile location of your files created in previous steps.

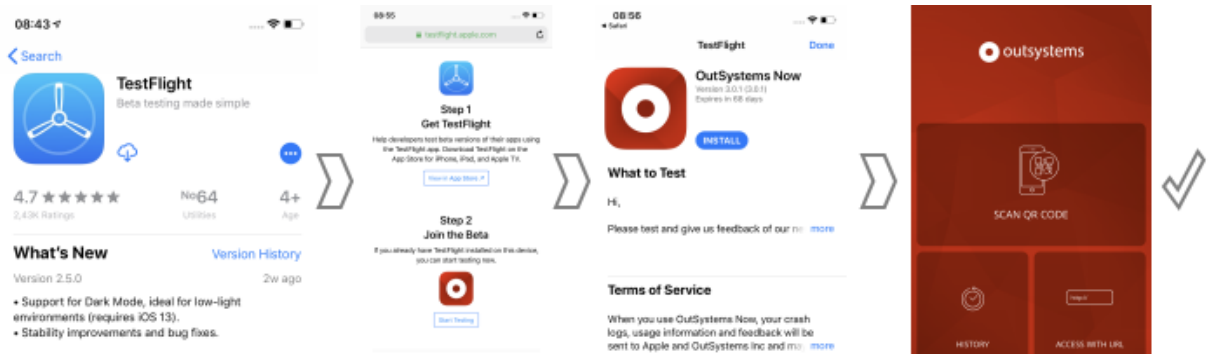


OutSytems will now generate the iOS App. Wait for the QR-code to appear.

6) Install and test Your Application on iOS devices

We are using Apple's test tool TestFlight by following these steps:

1. Install TestFlight on the iOS device that you'll use for testing.
2. Open this link on your iOS device.
3. Tap View in TestFlight or Start Testing.
4. Tap Install or Update for the app you want to test.



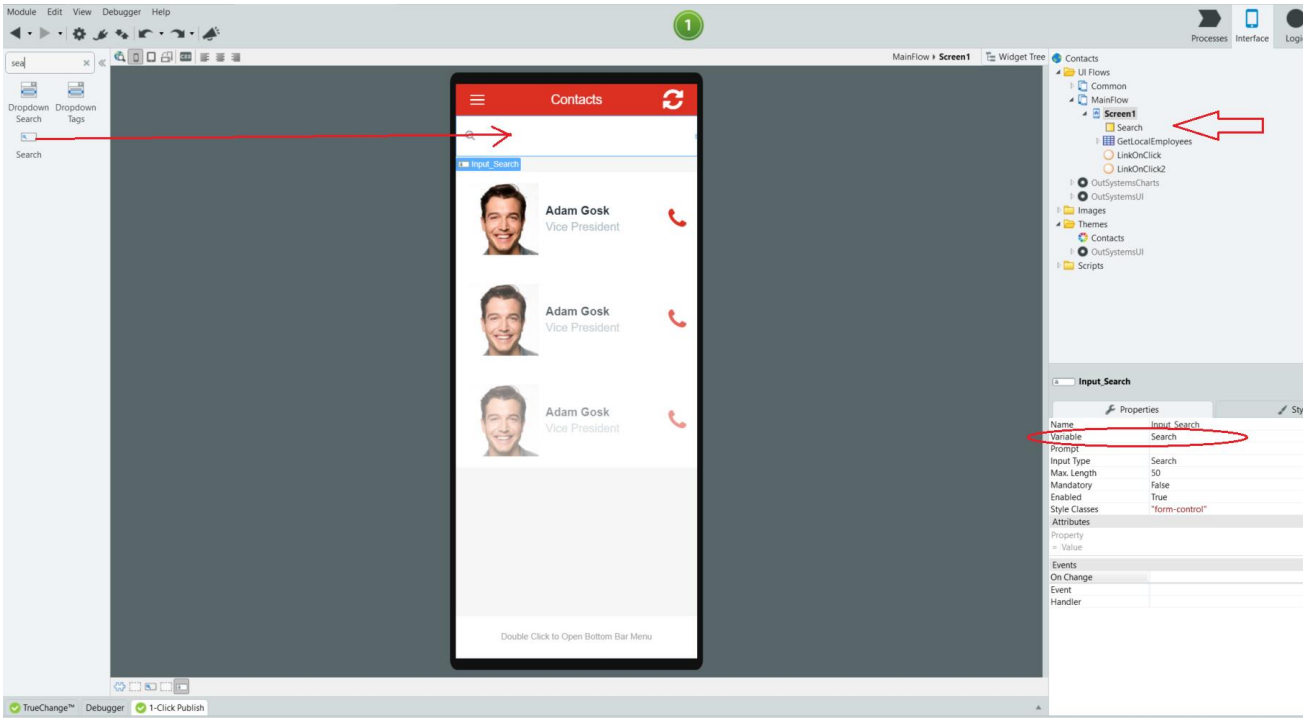
With the QR-code reader from **Outsystems Now** you can with your mobile device directly open the website where your application is stored. Otherwise copy the created link and go there with **Outsystems Now** on your mobile device.

10. Adding a search on the Contacts application

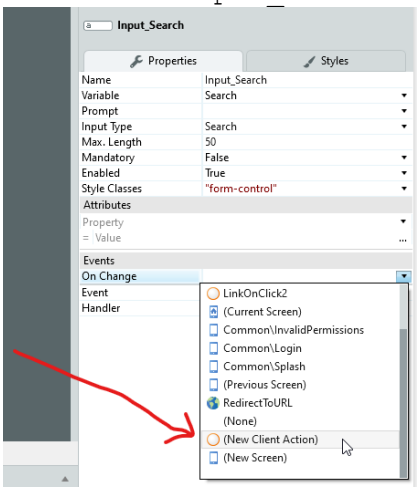
Go to the **Interface tab** of the **Contacts** Application. Select `Screen1` , right click and select the **Add Local Variable**. Name it `Search`.

From the left pane search for the **Search** widget and drag it to the Header of your application.

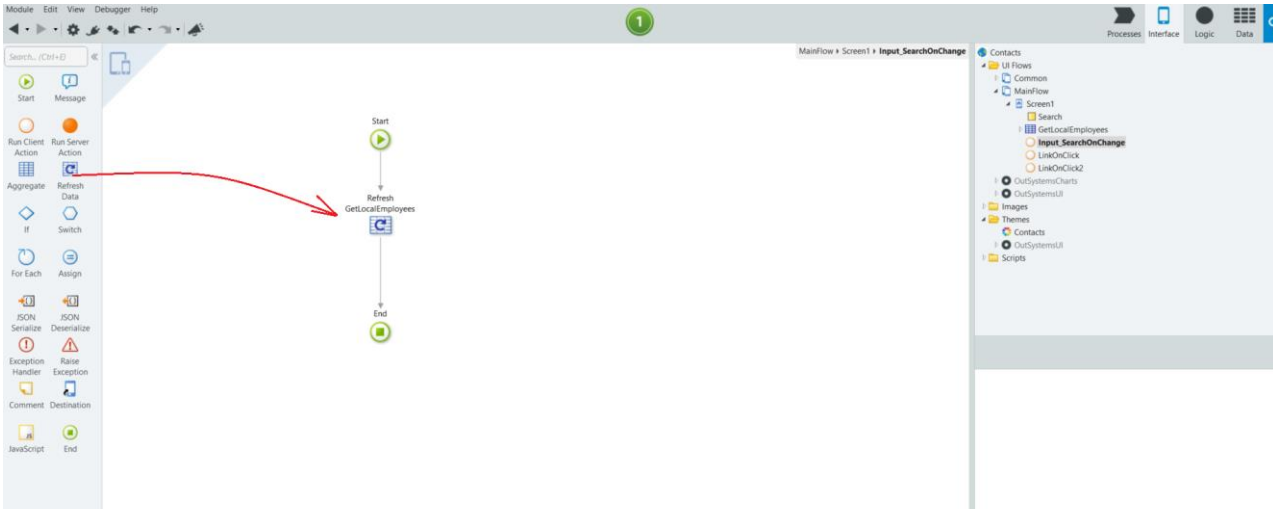
Add the **Local Variable** `Search` as input **variable** of the **Input_Search** widget.



On the **Input_Search** widget Properties pane, click on the **Events, On Change** right-side arrow to get the drop-down list and select the **New Client Action.** It results in a new Client Action called `Input_SearchOnChange` which also will be opened automatically.



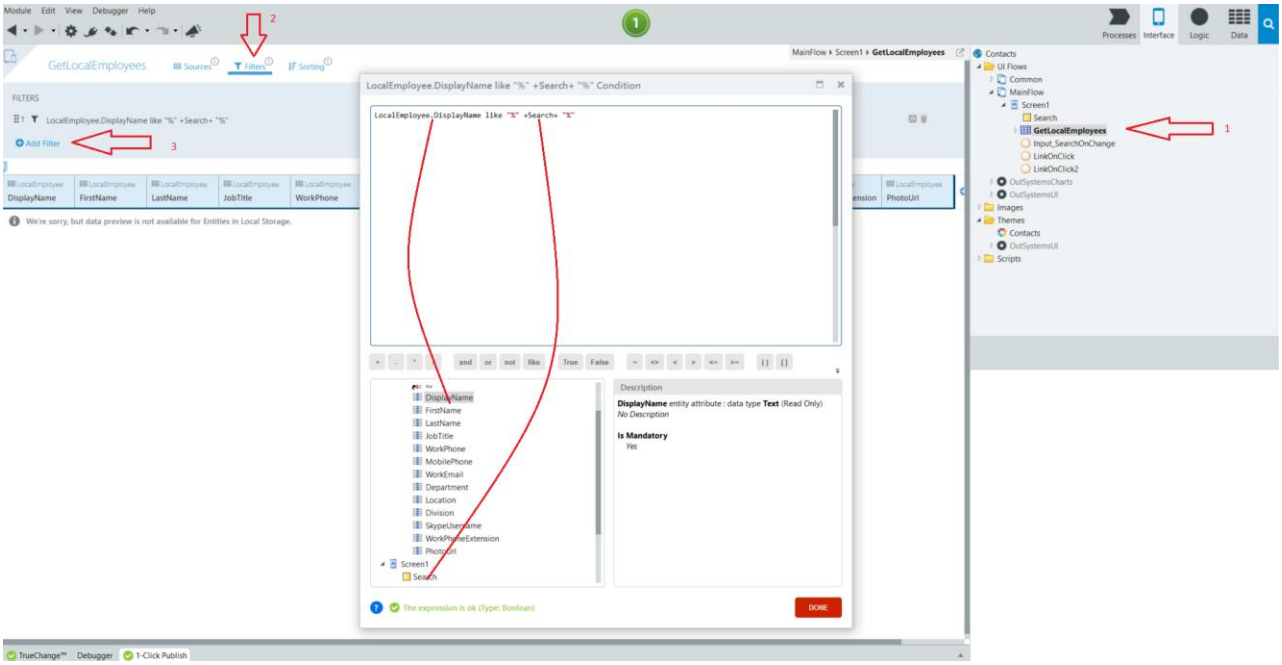
Drag and drop the **Refresh Data** icon from the left pane on your `Input_SearchOnChange` action.



Change the Aggregate `GetLocalEmployees` (double click) select **Filter** and **add a Filter**.

Select the `LocalEmployee.DisplayName` select `LIKE` and input `"%" +` select the local variable `search` and input a `+ "%".`

So it looks like: `LocalEmployee.DisplayName like "%" +Search+ "%"`



Now you are ready to test your application again by publishing and after that start the browser.

Check your application on your mobile device. Notice that this new version is automatic updated the old application and there is no need to reinstall your application.