

# Creating an Offline App with Local Storage

## Table of Contents

<b>Outline.....</b>	<b>2</b>
Scenario	2
<b>How-To.....</b>	<b>3</b>
Create Data Model	5
Display the List of Categories	8
Add Categories	11
Display Tasks	17
Add and Edit Tasks	18
Add Links to the Menu	24

# Outline

This exercise focuses on developing an app solely based on local storage. This app allows you to:

- Manage Categories
- List existing Tasks
- Create and Edit Tasks

When completed, the application is going to have three Screens and two Local Storage Entities that support managing tasks and categories.

## Scenario

In this exercise, start by creating a new application with one Module. You are going to create the Screens and Local Storage Entities inside this Module.

One of the Screens allows you to list existing categories and also add new ones using a text input and a Button.

Dedicate the remaining two Screens to tasks. One displays the list of tasks, while the other allows you to create new tasks or edit existing ones.

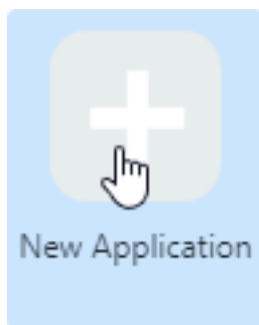
The application menu enables navigation between the different Screens.

# How-To

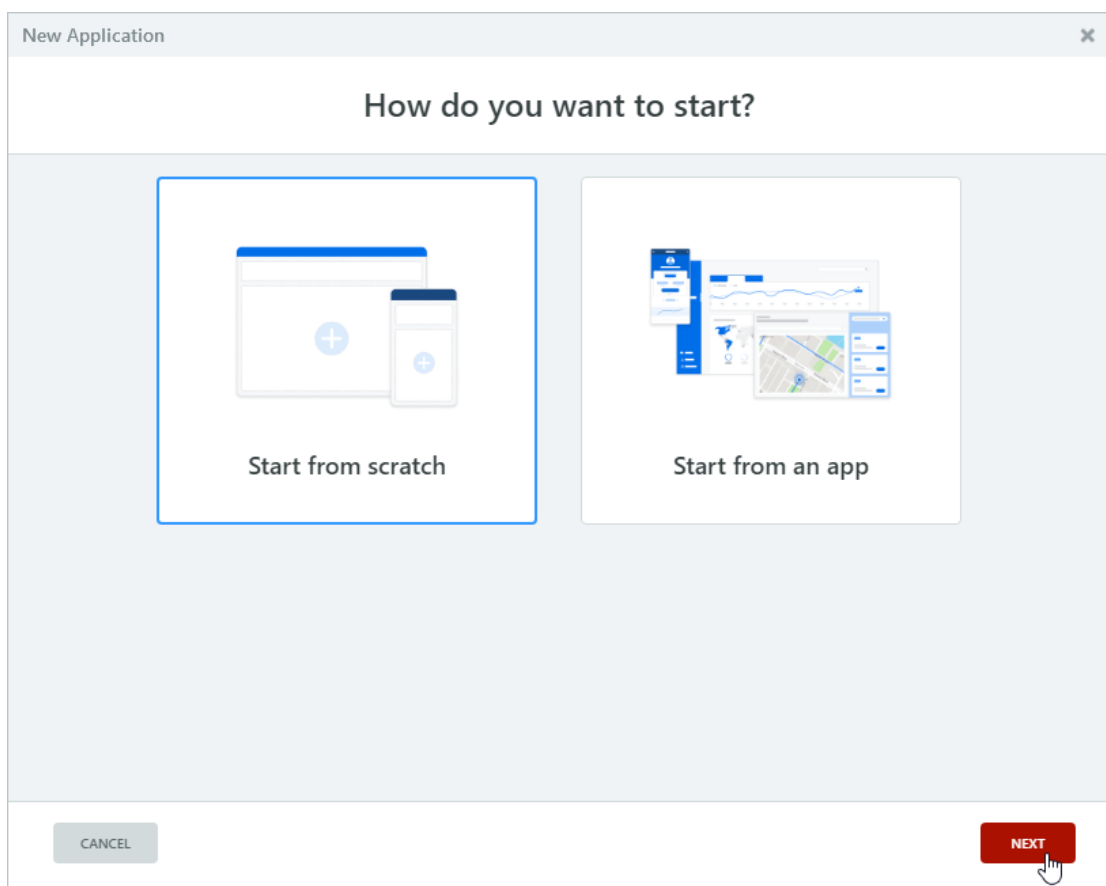
In this section, you are going to learn how to do this exercise, with a thorough step-by-step description. **If you already finished the exercise on your own, great! You don't need to do it again.** If you didn't finish the exercise, that's fine! This exercise is here to help you.

The first step is to create the Phone App and the respective Module.

- 1) In the Applications tab of Service Studio click **New Application**.

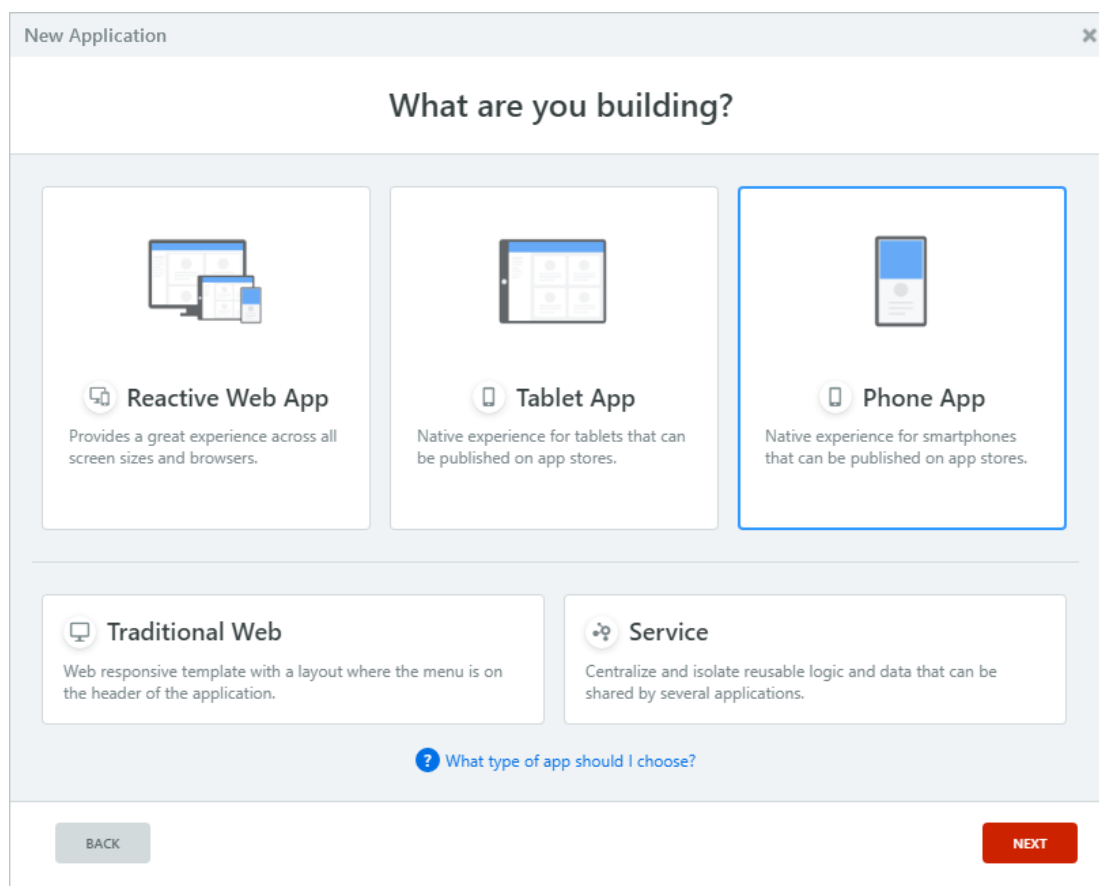


- 2) Select **Start from scratch** and click **Next**.



Depending on the version of Service Studio installed and the server you are connected to, this step may not appear. If that's the case, proceed to the next step.

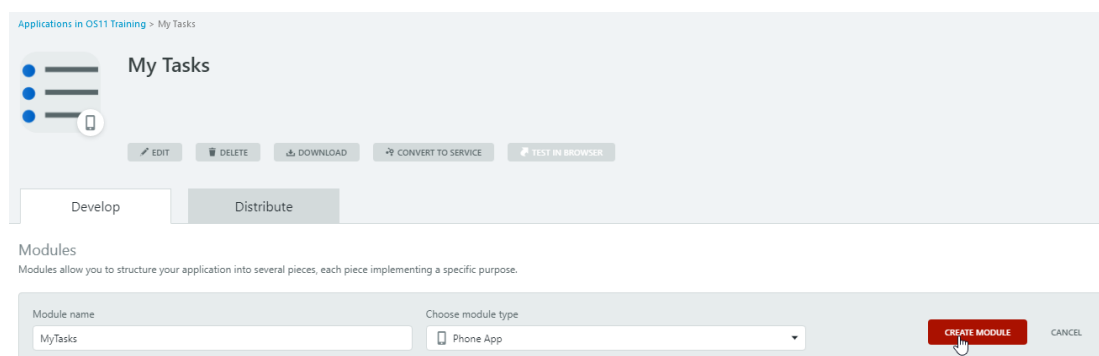
- 3) Select **Phone App** and click **Next**.



- 4) Set the name of the app to *My Tasks*, then click **Create App**.

Optionally, you may also add an icon and pick the a color of your choice.

- 5) Click **Create Module**.

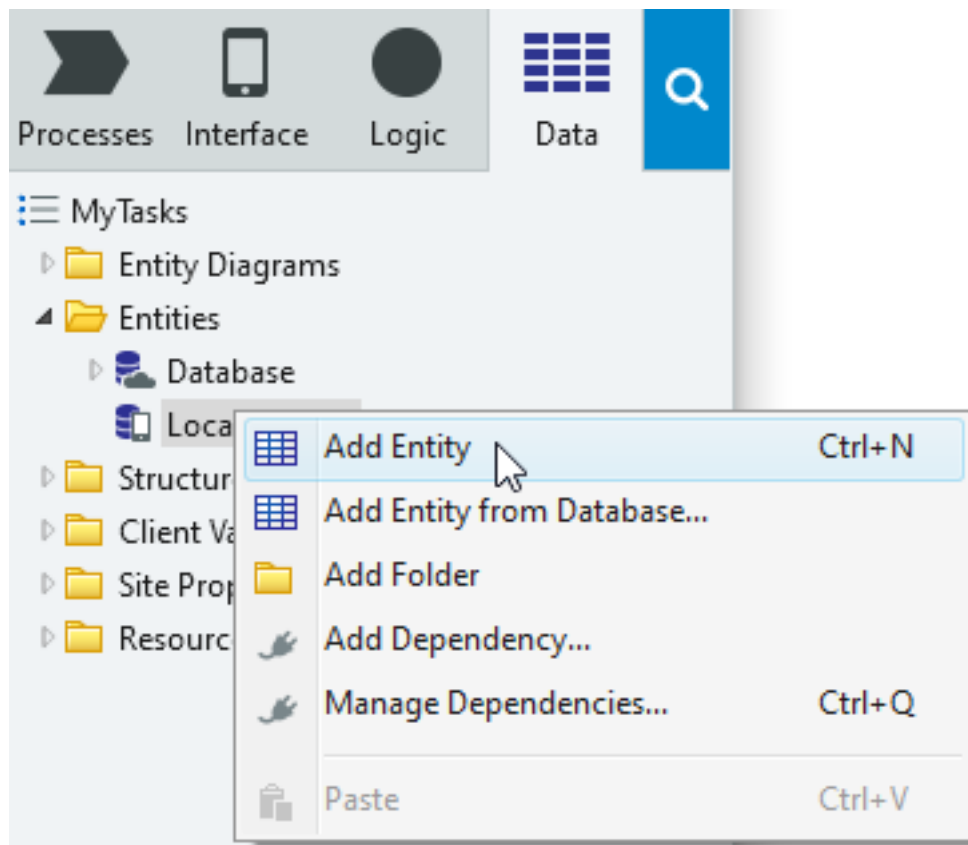


- 6) **Publish** the Module to save the current state.

## Create Data Model

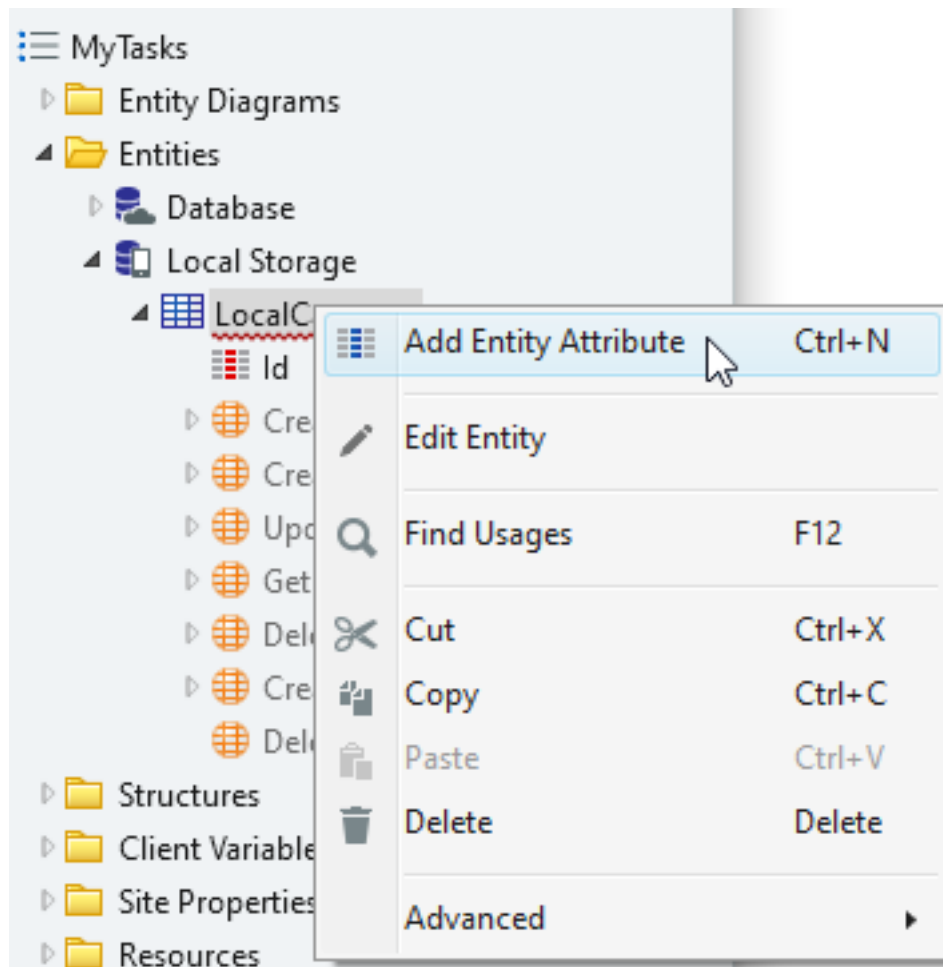
This section shows you how to create the local storage data model. You are going to use two Entities: one to store Categories, and another to store Tasks.

- 1) Switch to the **Data** tab.
- 2) Right-click the Local Storage folder and select **Add Entity**.



- 3) Set Entity **Name** to *LocalCategory*.

- 4) Right-click the newly created Entity and select **Add Entity Attribute**.



- 5) Set the **Name** of the attribute to *Name* and make sure the **Data Type** is set to *Text*.

Name Entity Attribute	
Name	Name
Description	...
Label	Name
Data Type	Text ▼
Length	50
Is Mandatory	No ▼
Default Value	...

- 6) Right-click the Local Storage folder and select **Add Entity**.
- 7) Set the **Name** of the Entity to *LocalTask*.

- 8) Add a new attribute to the Task Entity; name it *Description* with *Text* data type.

Description Entity Attribute	
Name	Description
Description	...
Label	Description
Data Type	Text ▼
Length	50
Is Mandatory	No ▼
Default Value	...

- 9) Add a new attribute and set its **Name** to *LocalCategoryId* and the make sure the **Data Type** is set to *LocalCategory Identifier*.

LocalCategoryId Entity Attribute	
Name	LocalCategoryId
Description	...
Label	Local Category
Data Type	LocalCategory Identifier ▼
Is Mandatory	No ▼
Delete Rule	Ignore

- 10) Add two more attributes: the *DueDate* with data type set to *Date Time*; and the *IsCompleted* with *Boolean* data type.

DueDate Entity Attribute	
Name	DueDate
Description	...
Label	Due Date
Data Type	Date ▼
Is Mandatory	No ▼
Default Value	...

IsCompleted Entity Attribute	
Name	IsCompleted
Description	...
Label	Is Completed
Data Type	Boolean ▼
Is Mandatory	No ▼
Default Value	▼

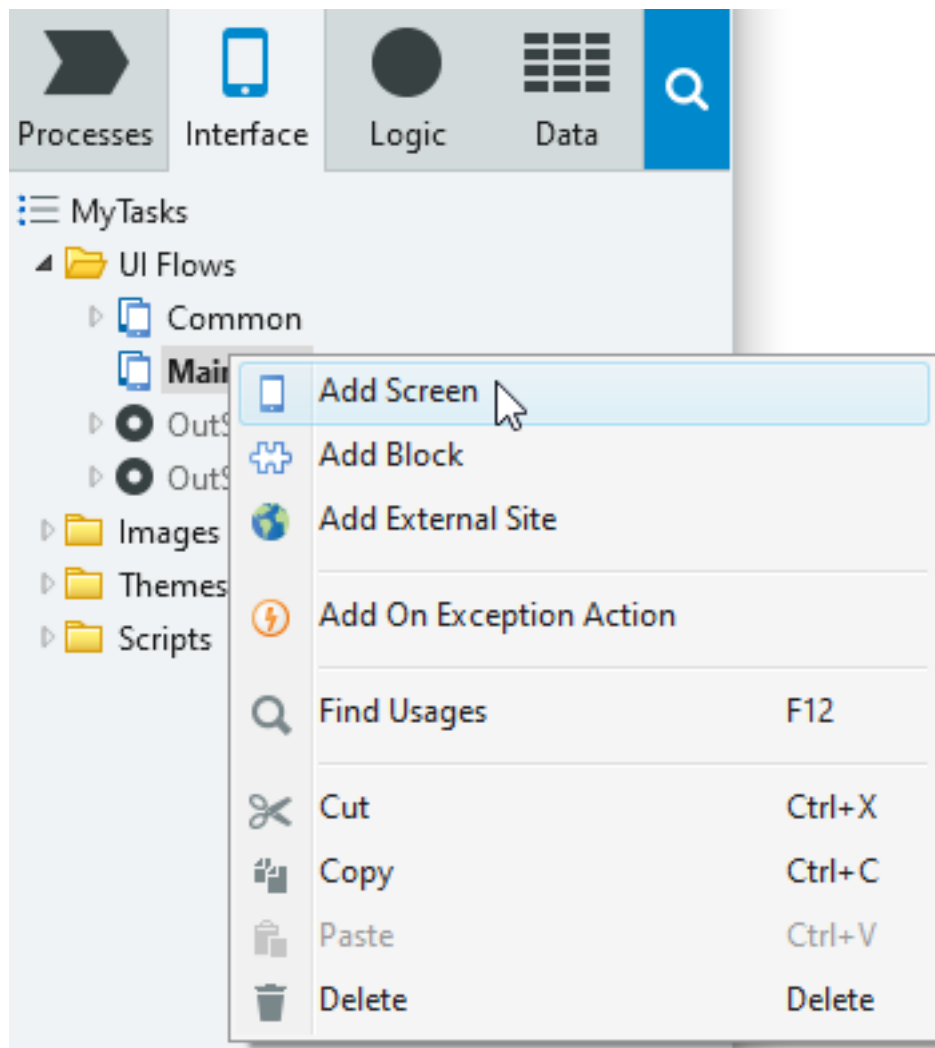
- 11) **Publish** the module.

At this moment, we have the Local Storage Entities created. As best practice, and following architecture guidelines, the Entities should be split apart from the user interface in different Modules. This ensures better maintainability of your apps.

## Display the List of Categories

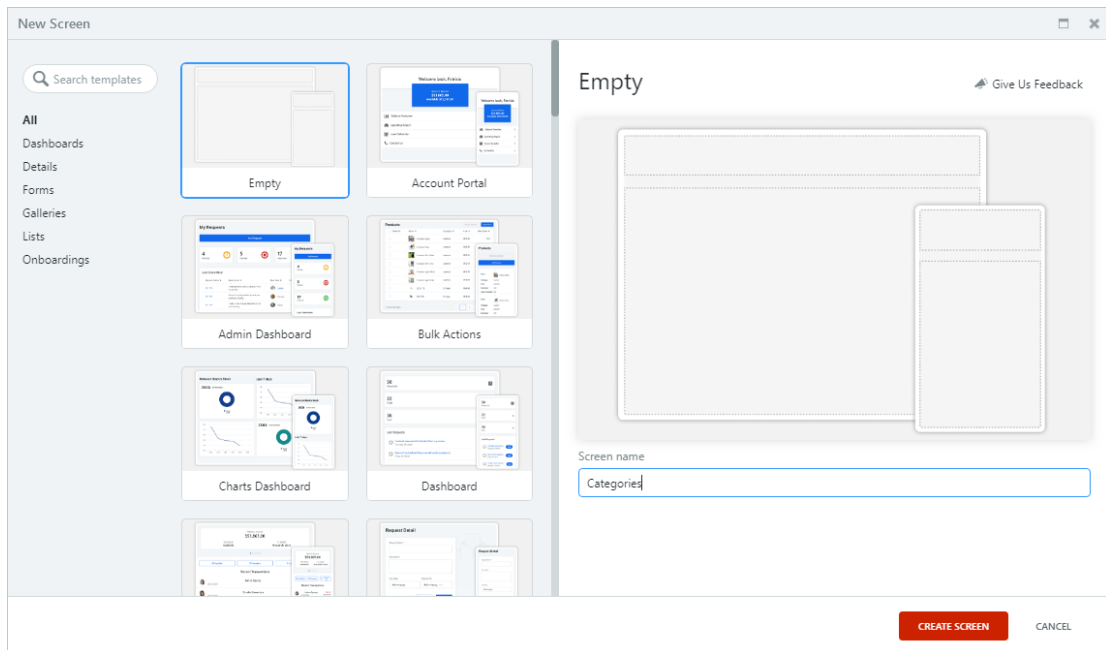
At this point, we have the Local Storage Entities created. It is now time to start developing the user interface to display the data from the LocalCategory Entity.

- 1) Switch to the **Interface** tab.
- 2) Right-click the *MainFlow* and select **Add Screen**.

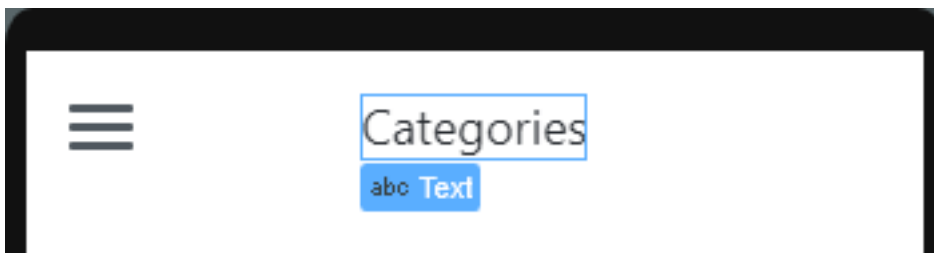




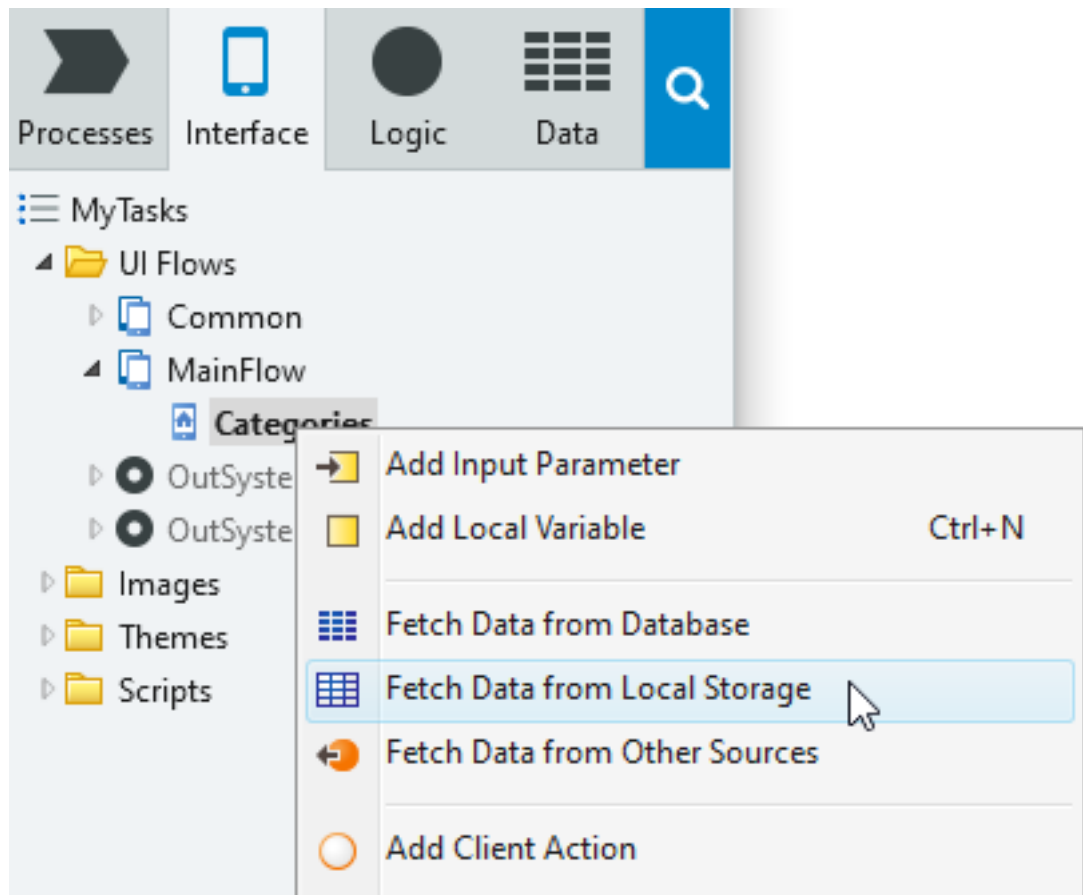
- 3) Select the **Empty** screen template, set the Name of the screen to *Categories* and then click **Create Screen**.



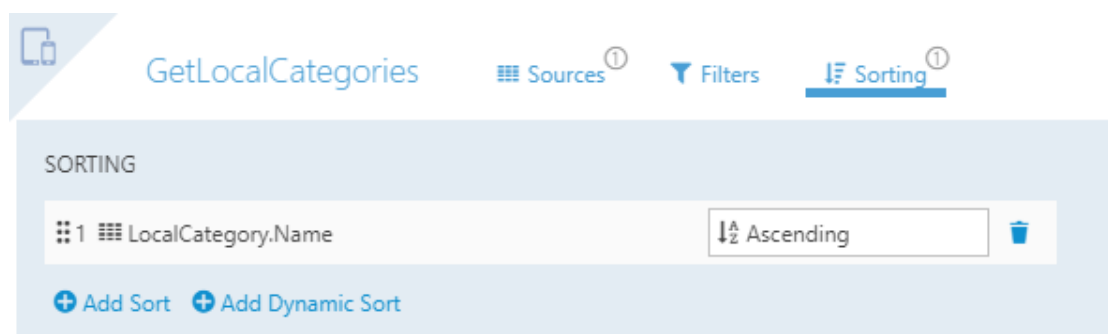
- 4) Click on the Title placeholder and type *Categories* inside it.



- 5) Right-click the Categories Screen and select **Fetch Data from Local Storage**.

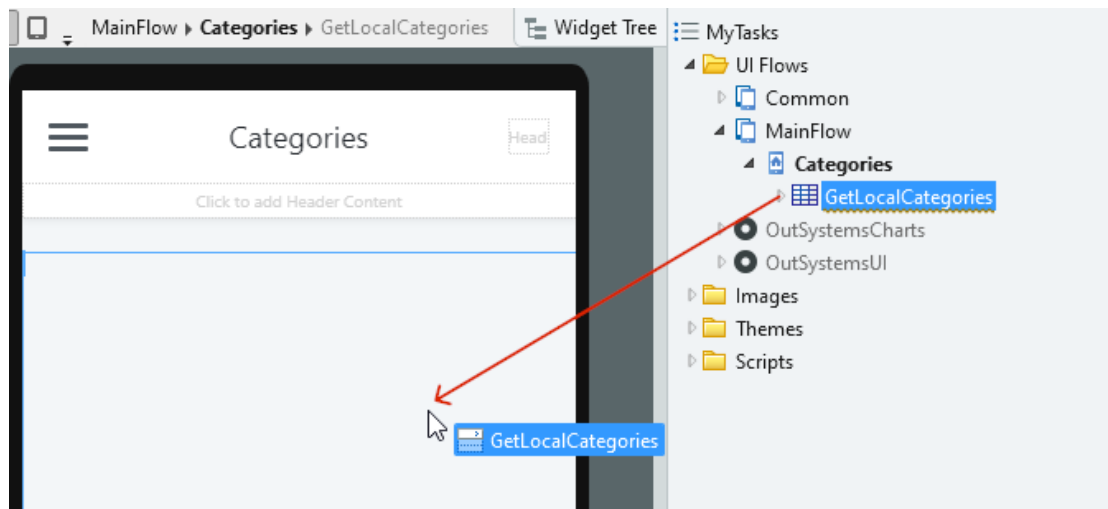


- 6) Add the **LocalCategory** Entity to the Aggregate by clicking it and selecting the Entity from the popup dialog.
- 7) On the Sorting tab, click on **Add Sort** and add a new criteria to sort by the category name.



- 8) Return to the Categories Screen.

- 9) Drag the GetLocalCategories Aggregate and drop it on the Screen.

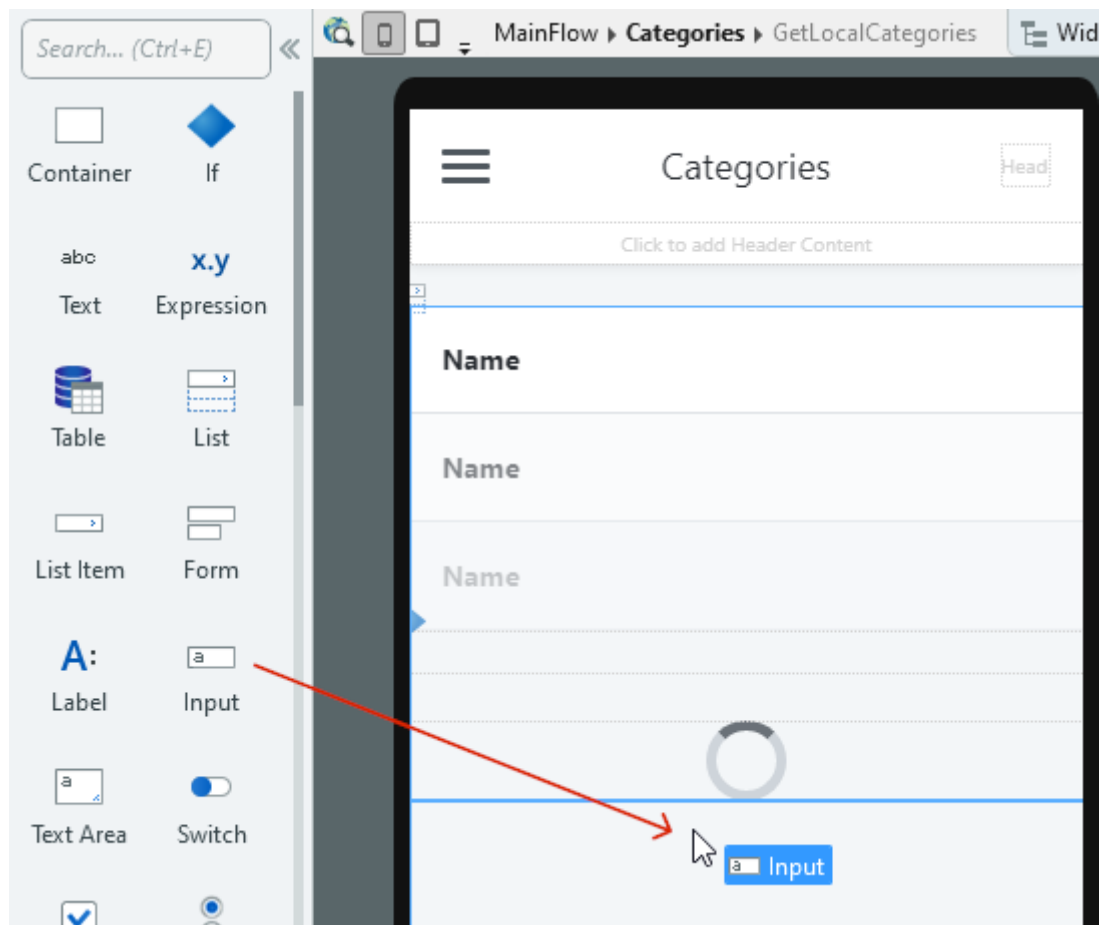


## Add Categories

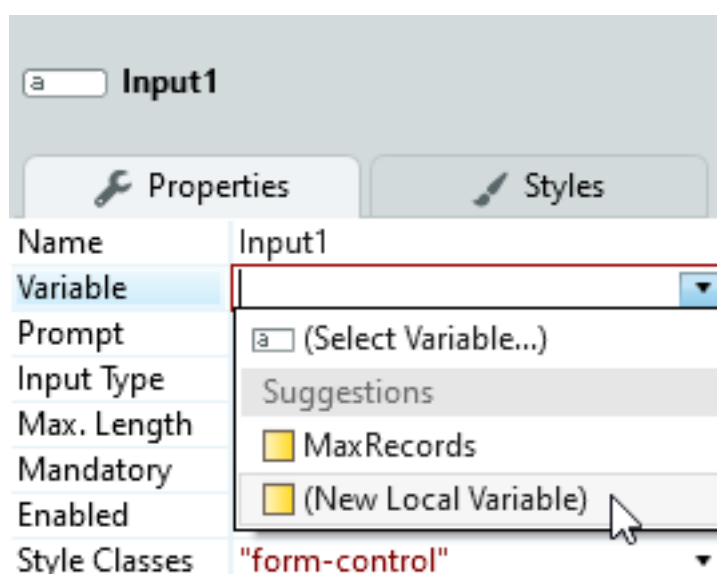
We'll now add a simple form that enables users to add new categories to the Local Storage Entity. The form will contain an input for the category name, and a Button that

will be bound to a Client Action. Inside the Client Action we will add the new category to the Local Storage Entity.


- 1) Drag an Input and drop it below the list created in the previous section.



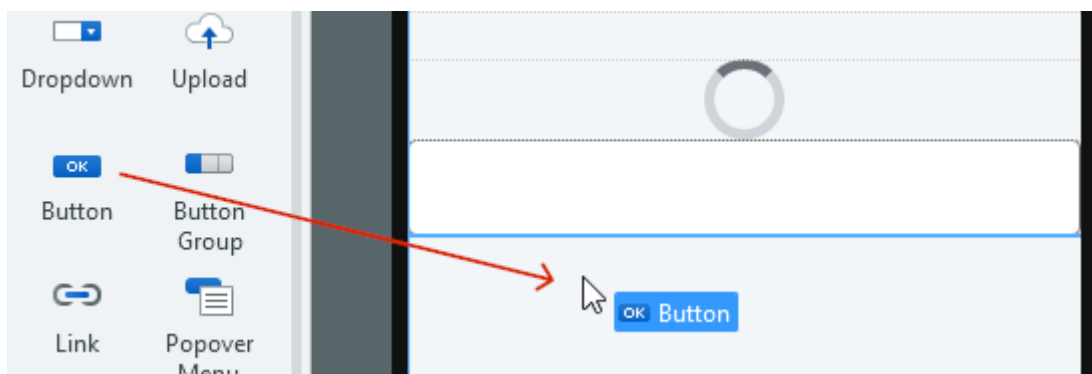
- 2) In the Variable property, open the dropdown of suggestions and select **(New Local Variable)**



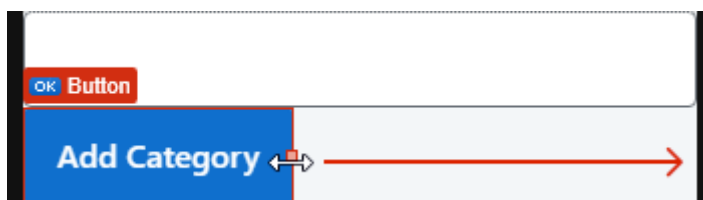
- 3) Rename the variable to *NewCategoryName*.

	<b>NewCategoryName</b> Local Variable
Name	NewCategoryName
Description	...
Data Type	Text ▼
Default Value	...

- 4) Drag a Button and drop it below the Input.



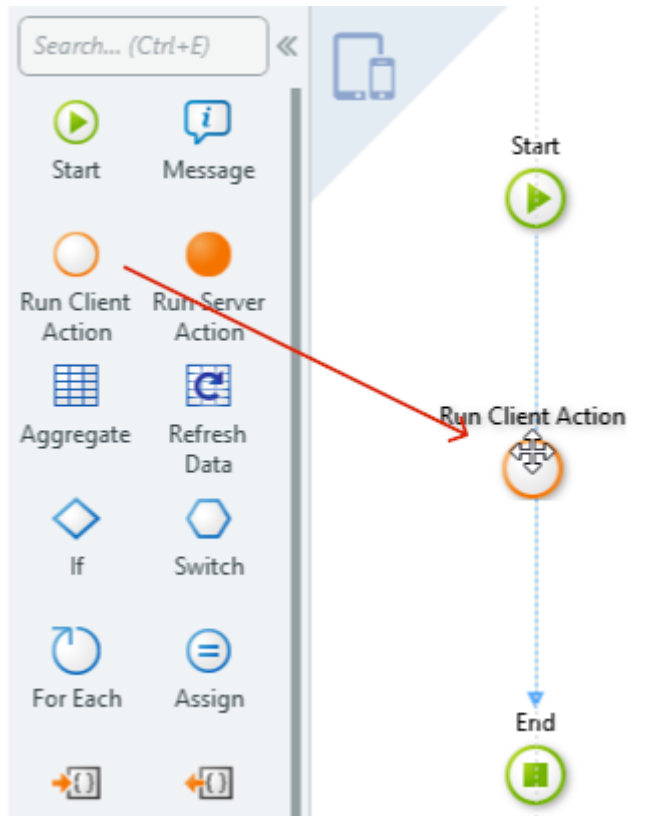
- 5) Change the text inside the Button to *Add Category*.
- 6) Select the Button and change its width to the full width of the Screen.



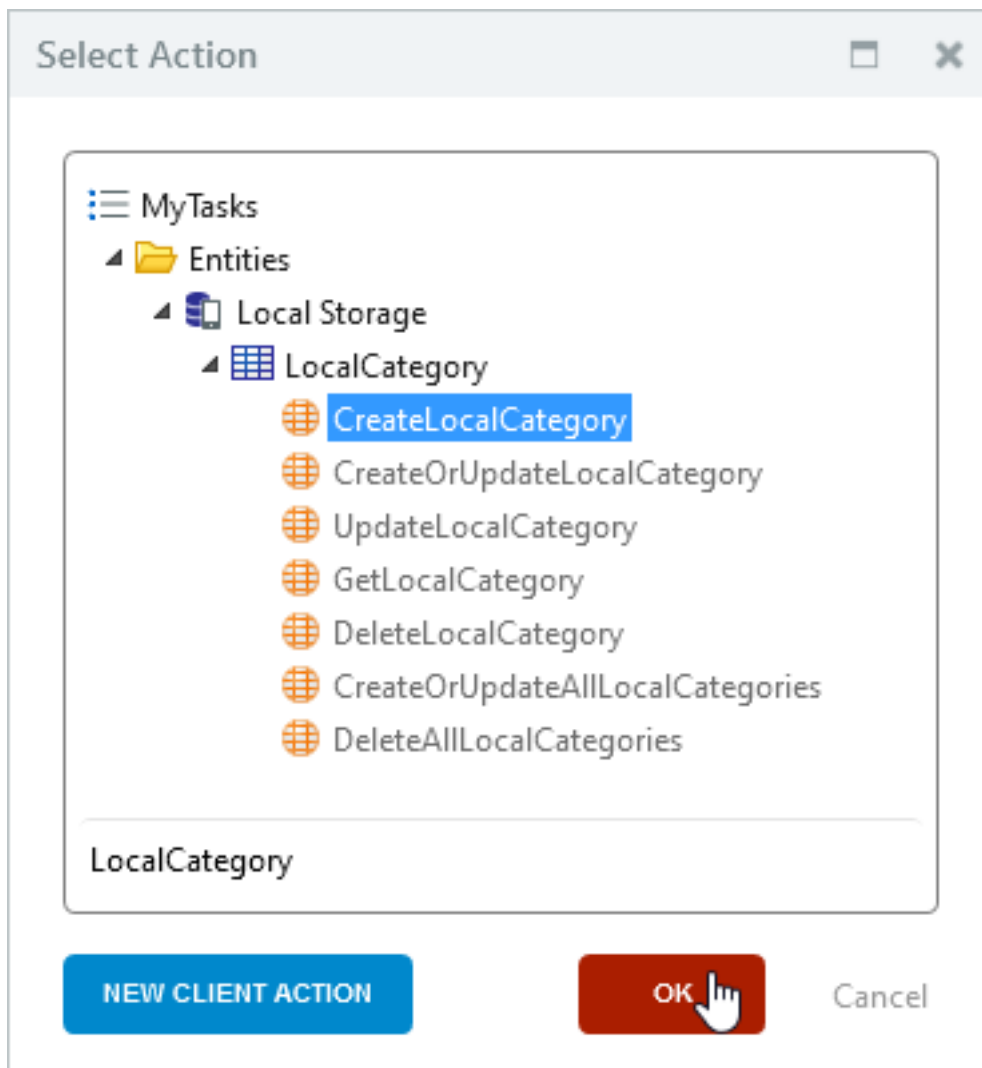
- 7) Double-click the Button to create the Client Action and bind it to the On Click property.

Another possibility to achieve the same output is to open the suggestions dropdown of the On Click property, and then select **(New Client Action)**.

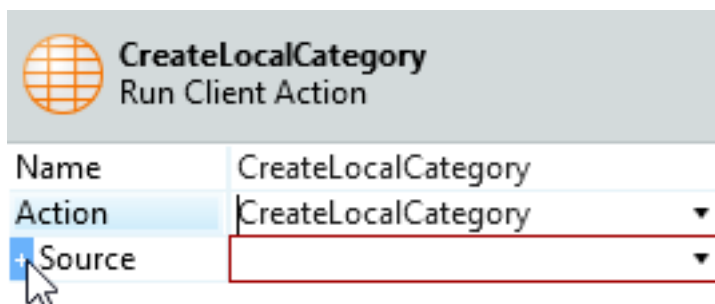
- 8) Drag a **Run Client Action** and drop it between the Start and End.



- 9) In the Select Action popup dialog select the **CreateLocalCategory** Entity action, then click **OK**.




- 10) In the Properties pane, click the Expand Source plus icon.



The Expand Source option allows you to create an inline record. This allows for a clear code, since it is not required to create an extra variable of type Record, assign its attributes, and only then use it as parameter to the Entity Action.

- 11) Set the **Id** attribute to `NullIdentifier()` and the **Name** to the *NewCategoryName* local variable.

 <b>CreateLocalCategory</b> Run Client Action	
Name	CreateLocalCategory
Action	CreateLocalCategory ▼
Source	
Id	NullIdentifier() ▼
Name	NewCategoryName ▼

**Challenge:** Add some form validations to the logic and to the Screen. For instance, validate that the category name has three or more characters.

- 12) Drag an Assign and drop it between the Entity Action and the End, then define the following assignment:

```
NewCategoryName
= ""
```

Although this step is not mandatory, it will make sure that after adding the category to the Local Storage Entity, the Input on the Screen is cleared.

- 13) Drag a Refresh Data and drop it between the Assign and the End. In the popup dialog, select the GetLocalCategories Aggregate.



14) The flow should look like this:



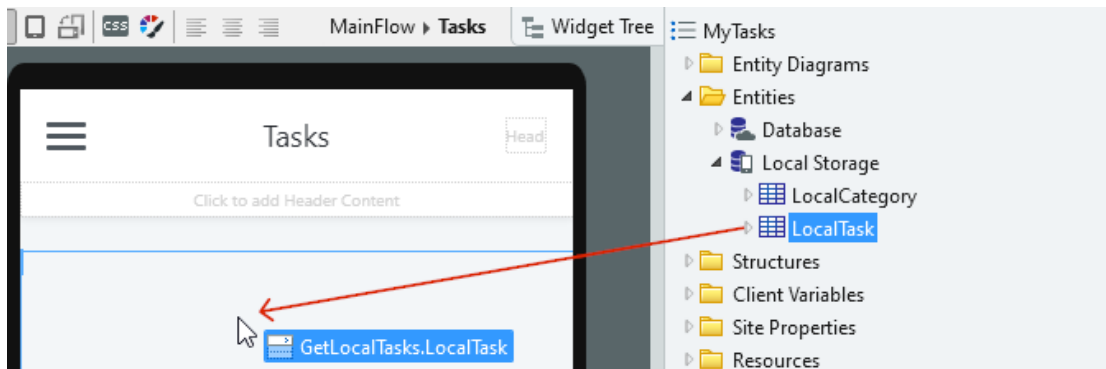
15) Publish the Module and test, adding a few categories of your choice.

## Display Tasks

In this section, we'll create another Screen in our Mobile App. This Screen, the Tasks Screen, will enable us to see the existing tasks. Later on, this Screen will enable end-users to navigate to a third Screen to create new tasks.

- 1) Right-click the MainFlow, and select **Add Screen**.
- 2) Select the **Empty** Screen template, and name it *Tasks*.
- 3) Inside the Title placeholder add the following text: *Tasks*

- 4) From the **Data** tab, drag the **LocalTask** Entity and drop it in the Content placeholder of the Screen.



This accelerator will create a few elements for you.

An Aggregate (GetLocalTasks) is created at the Screen level. This Aggregate has the LocalTask as source, and also the LocalCategory and the join between both Entities. It also has a sorting criteria by the task description.

On the Screen, a List Widget is added; its source property is set to the Aggregate created. Inside the list, several Widgets were also added, namely a List Item, and Expressions.

All these elements can be customized to meet your own needs.

- 5) Publish the Module to save the current state.

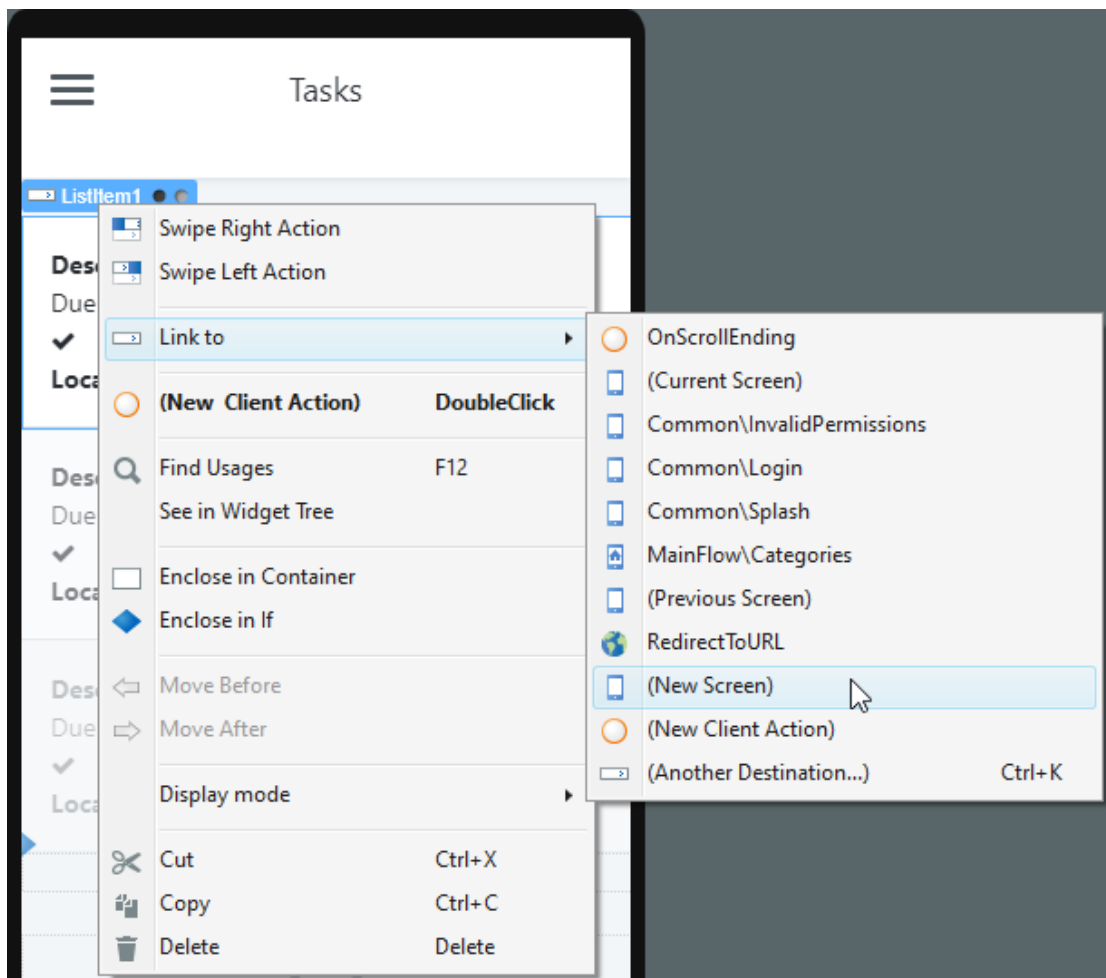
At this point, we don't have any tasks created yet, so we won't be able to properly test this Screen.

## Add and Edit Tasks

In this section, our app will be extended with the capability to add new tasks. We'll start by creating a new Screen and linking to it from the Tasks Screen. Inside the new Screen,

a form will be created with the Input fields that will then be used to insert a new record in the LocalTask Entity, or edit an existing one.

- 1) Select the List Item Widget, then right-click it and select **Link to > (New Screen)**. Name it *TaskDetail*.

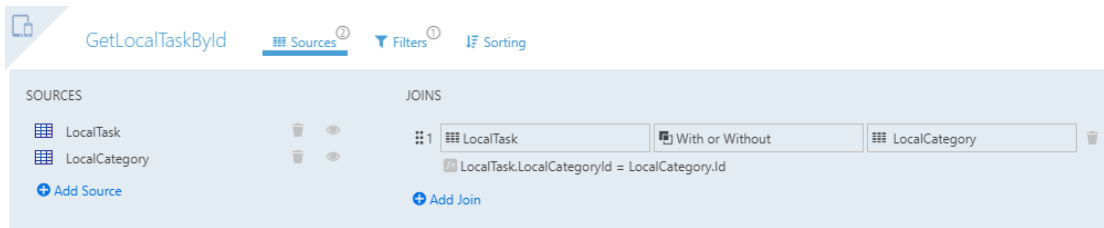


The new Screen will already have an Input parameter named LocalTaskId of type *LocalTask Identifier*. This parameter will be used to identify the task to be edited.

- 2) Right-click the TaskDetail Screen and select **Fetch Data from Local Storage**.
- 3) Drag the LocalTaskId Input parameter of the Screen and drop it inside the Aggregate canvas.

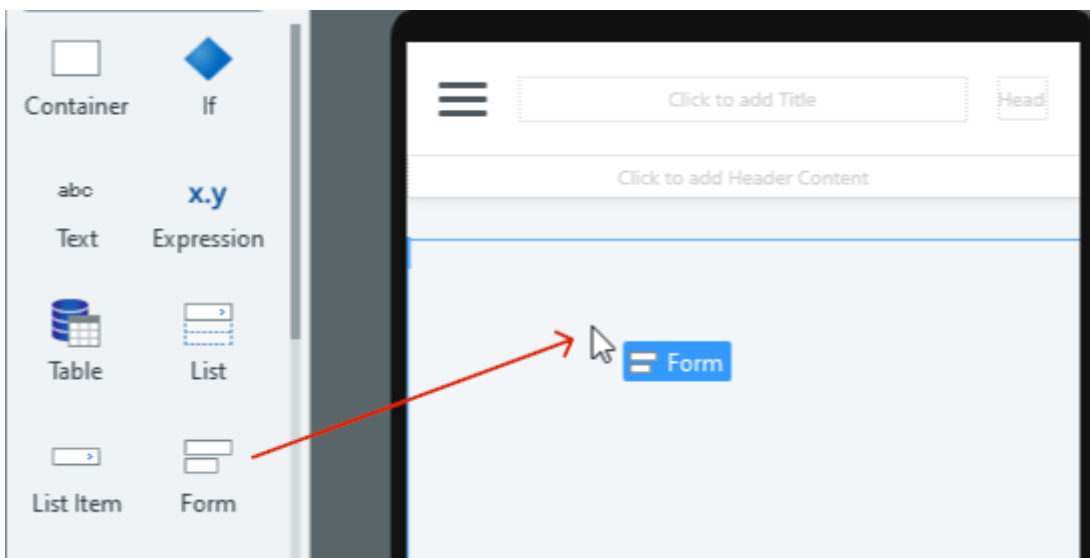
This accelerator automatically adds the LocalTask Entity as source, and creates a filter based on the LocalTask Id attribute and the LocalTaskId parameter.

- 4) In the Sources tab, add the LocalCategory Entity.

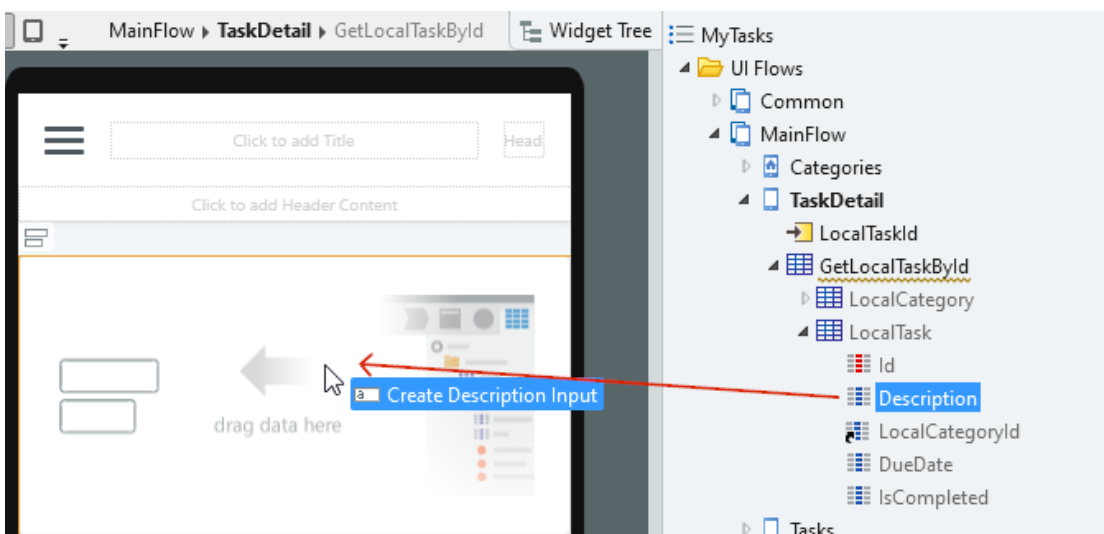


The join between the two Entities should be created automatically.

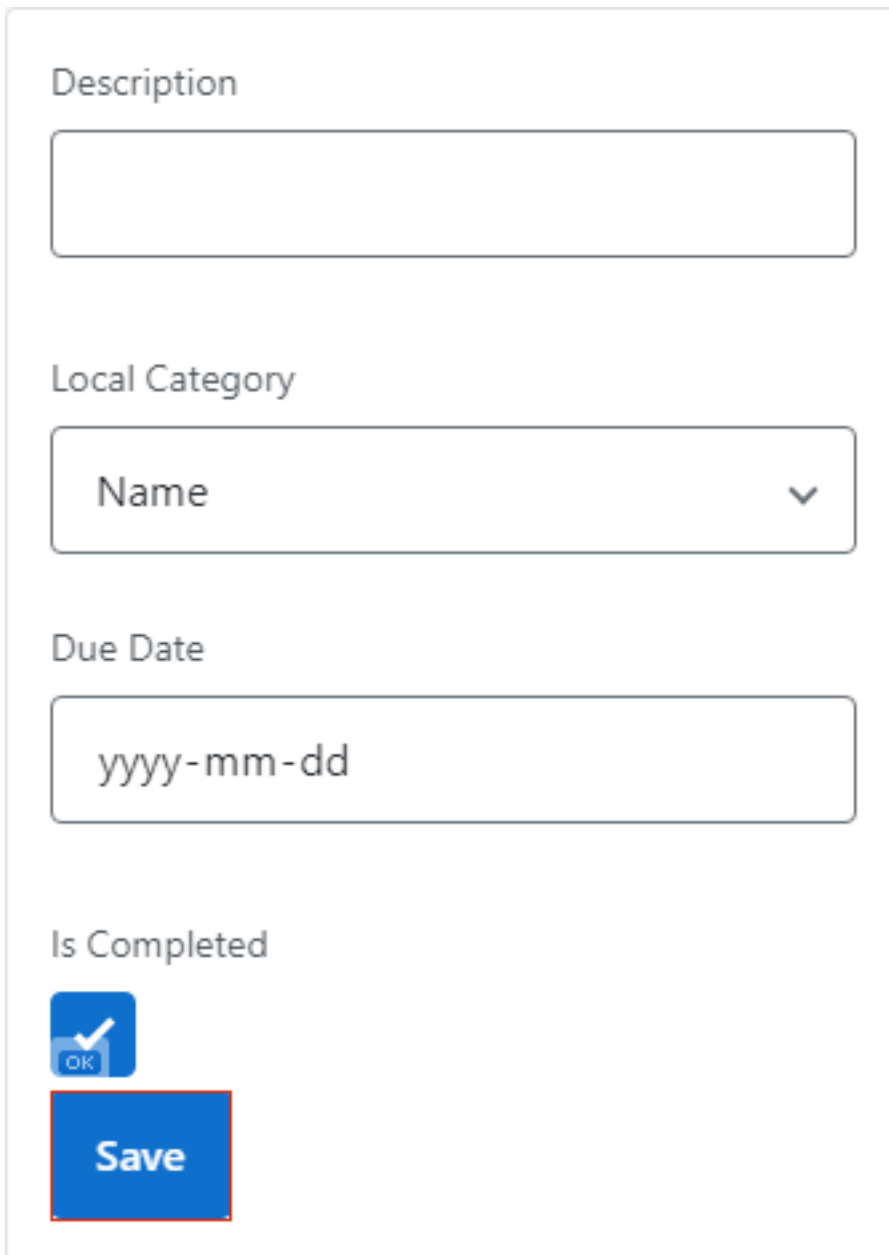
- 5) Return to the TaskDetail Screen.  
6) Drag a form and drop it inside the content area of the Screen.



- 7) Drag the Description attribute located under the GetLocalTaskById Aggregate and drop it inside the form.



- 8) Drag the LocalCategoryId, DueDate, and IsCompleted and drop just before the Save Button. The form should look like this:

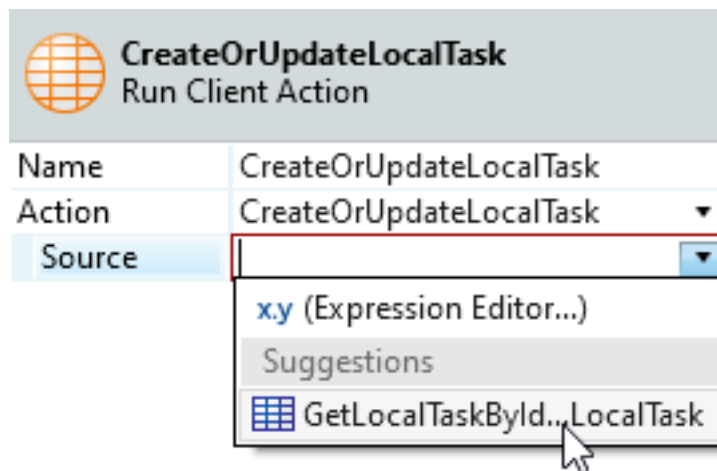


The screenshot shows a form layout within a light gray border. It contains the following elements from top to bottom:

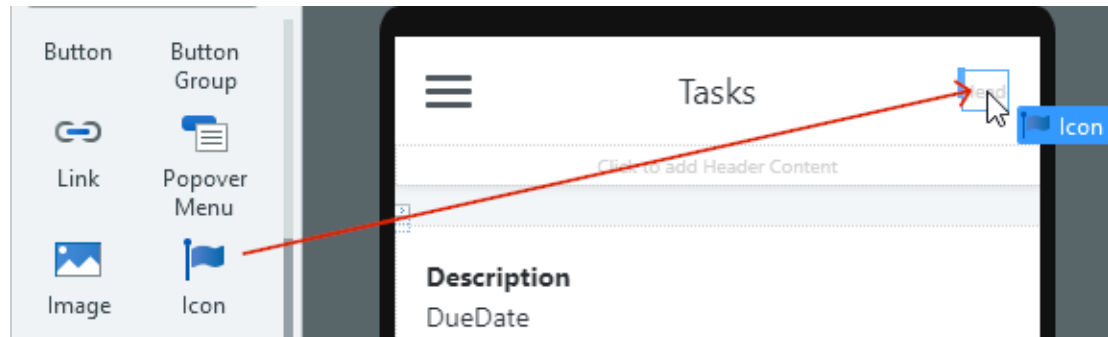
- A label "Description" above a large, empty rectangular text input field.
- A label "Local Category" above a dropdown menu. The dropdown menu has the text "Name" and a downward-pointing chevron icon.
- A label "Due Date" above a rectangular text input field containing the placeholder text "yyyy-mm-dd".
- A label "Is Completed" above a small blue square button with a white checkmark and the text "OK".
- A large blue rectangular button with the text "Save" in white, which is highlighted with a red border.

- 9) Change the Label of the Local Category Input to *Category*.
- 10) Double-click the Save Button to create its Client Action.
- 11) Drag a **Run Client Action** and drop it between the existing If and the End.
- 12) In the popup dialog, select the CreateOrUpdateLocalTask Entity Action.

- 13) Set the source property of the Entity Action to `GetLocalTaskById.List.Current.LocalTask`.

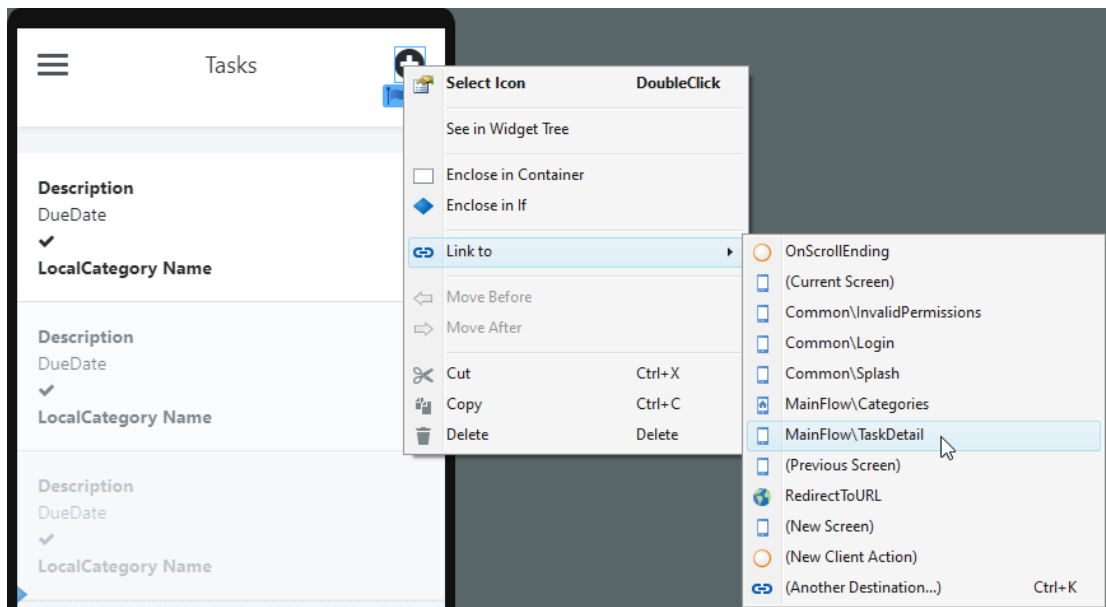


- 14) Drag a Destination and drop it on top of the End to replace it, then select the Tasks Screen as the Destination Screen.
- 15) Open the **Tasks** Screen.
- 16) Drag an **Icon** and drop it in the **HeaderRight** placeholder.

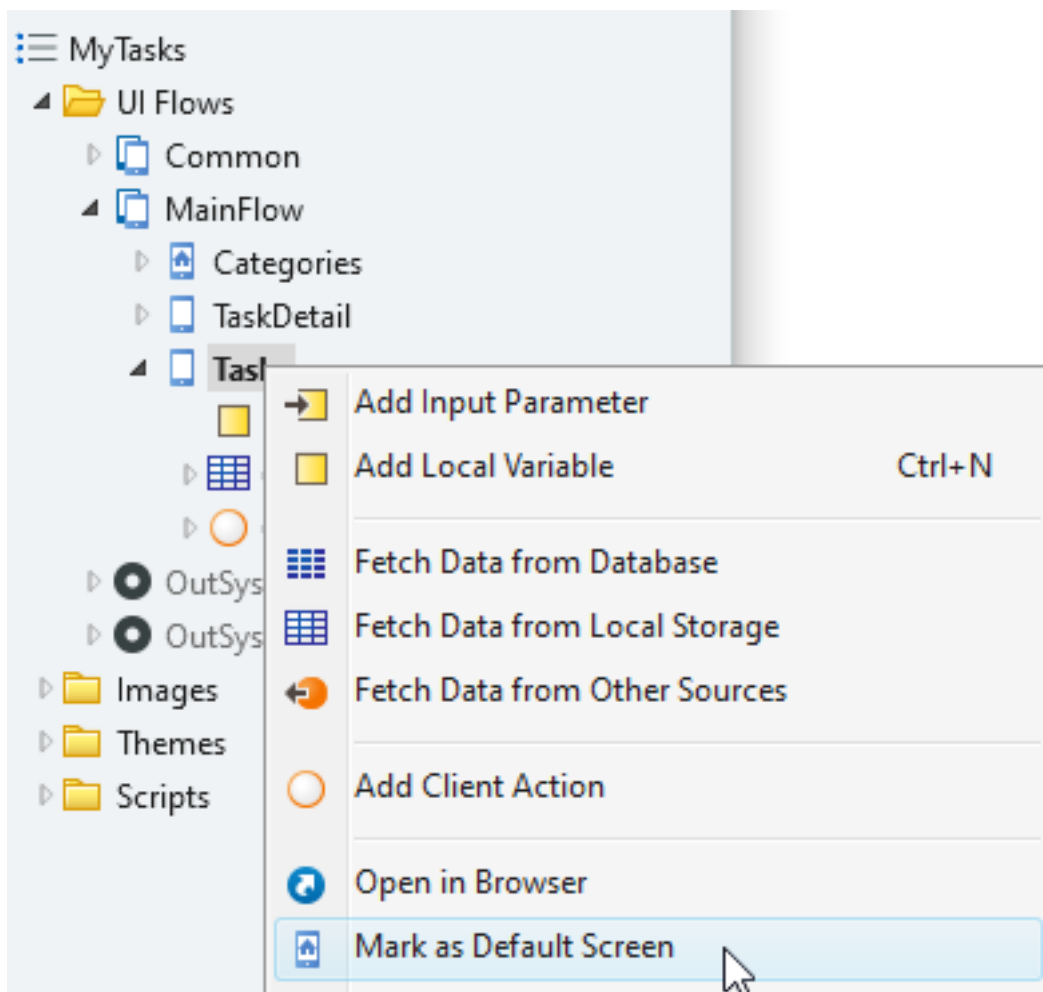


- 17) In the Pick an Icon dialog, select the *plus circle* icon.

18) Right-click the Icon and select **Link to > TaskDetail**.



19) Right-click the Tasks Screen and select **Mark as Default Screen**.



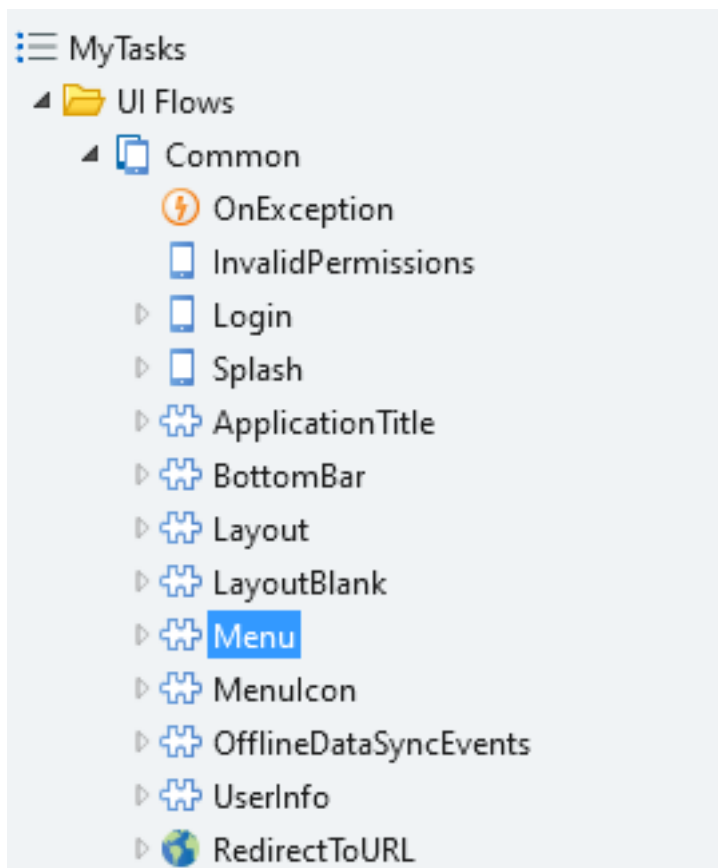
20) Publish the Module, then click Open in Browser.

21) Add a few tasks to your app.

## Add Links to the Menu

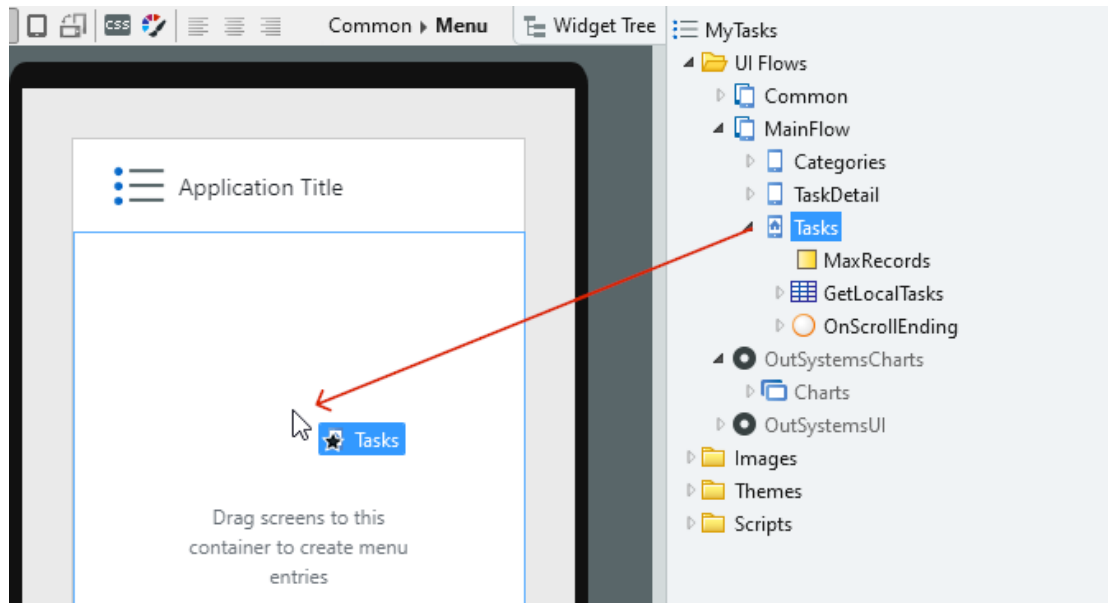
At this point, you should have three Screens: One that allows you to manage the categories, one to display the tasks, and another one to create or edit tasks. Since the Tasks Screen is now the default Screen and there is no link to the Categories Screen, there is actually no way for an end-user to reach that Screen. We'll now add links in the menu to enable navigation between the different Screens.

1) Under the Common UI Flow, open the Menu Block.

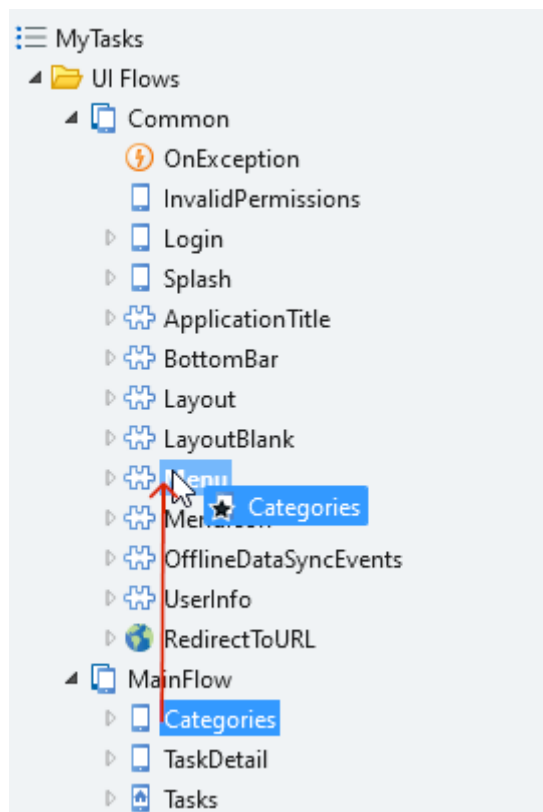




- 2) Drag the Tasks Screen and drop in the center of the Menu Block.



- 3) Drag the Categories Screen and drop it on top of the Menu element in the tree.



- 4) Publish the Module and test the navigation to the different Screens using the menu.