AWSUG Bengaluru

CLI Commands **2**

Prerequisites **3**

AWS 3

AWS CLI 3

Windows: 3

Mac: 3

Outsystems environment 4

Demo **5**

Summary 5

Demo Steps 5

Starting with the AWS CLI 5

Showing the data in Service Studio 8

Creating a detail Screen 10

Adding UI to take pictures 22

Rekognition + S3 25

Finishing up 31

CLI Commands

Installing AWS CLI Windows:

1 - Run in the CMD the following command:
msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
Follow the wizard to install, then close and open the CMD

2 - Run CLi Commands for DynamoDB:

Scan the **Restaurant** db
aws dynamodb scan --table-name Restaurant

Get Specific Item (windows version with escaped “)
aws dynamodb get-item --table-name Restaurant --key
{\"Id\":{\"S\":\"638f3b19c2b0fb839b3652b8\"}}

Get Specific Item
aws dynamodb get-item --table-name Restaurant --key
‘{"Id":{"S":"638f3b19c2b0fb839b3652b8"}}’

3- Run Commands for S3:

List content of os.restaurant-images
aws s3 ls s3://os-restaurant-images --recursive --human-readable --summarize

Copy file “**testkey**” from the bucket “**os-restaurant-images”** to disk:
aws s3 cp s3://os-restaurant-images/testkey .

List buckets
aws s3api list-buckets --output json

Prerequisites

AWS

You need to have created:

- ☐ One bucket on S3 with the name **os-restaurant-images**
- ☐ One DynamoDB Table with the name **Restaurant** populated with the data from this JSON
- ☐ Access to Rekognition
- ☐ An IAM user configured with access to these 3 services
- ☐ Make sure that the S3 and the DynamoDB table are set in the same region

AWS CLI

You need to install the AWS CLI on your machine and configure the user to your IAM one

Windows:

1 - Open a CMD and run the command **msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi** (run installer)
2 - Follow the instructions on the installer until is finished
3 - Close the CMD and open a new one (so it refreshes and has the aws cli installed)
4 - Run the command **aws configure**. Set the Access Key Id, the Access Key and the Region
5 - You are all set to use the CLI

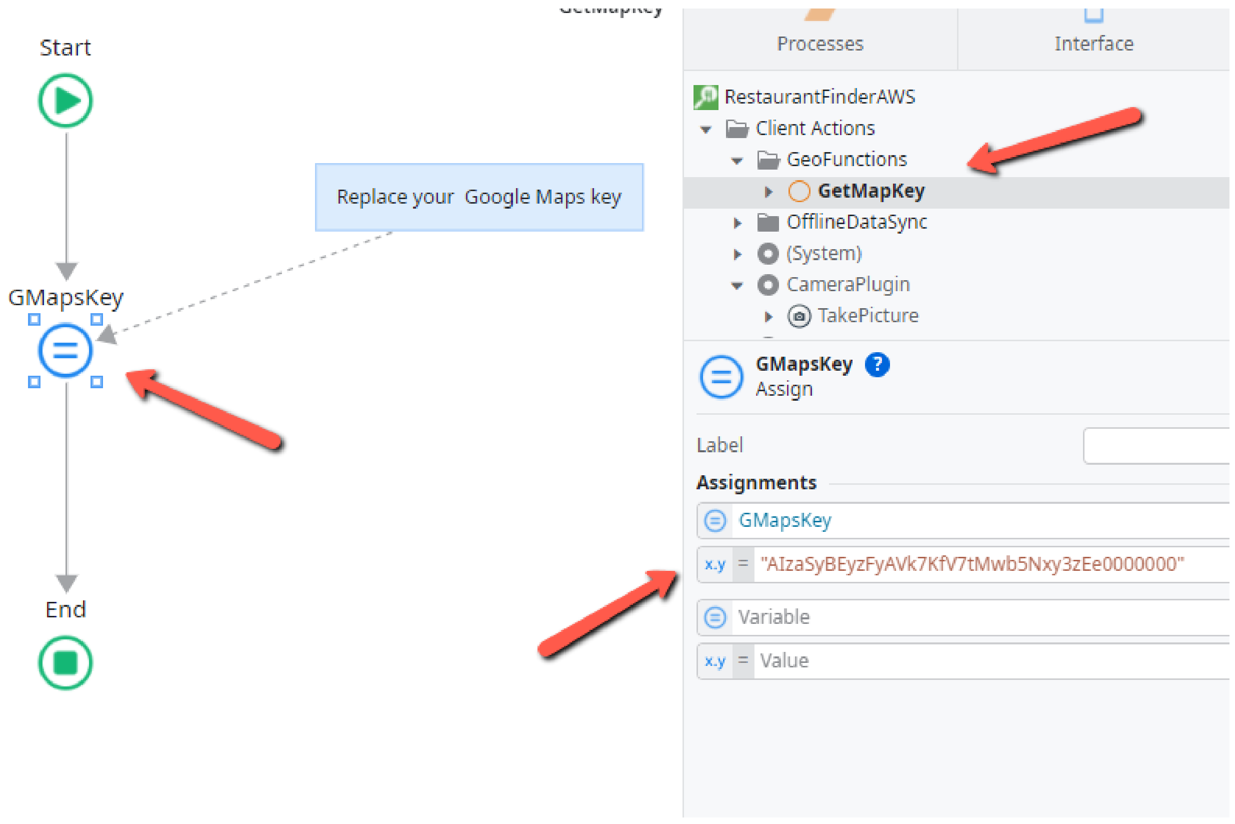
Mac:

1 - Download and install https://awscli.amazonaws.com/AWSCLIV2.pkg
2 - Open the Terminal and check if it's working using the command **aws --version**
3 - Run the command **aws configure**. Set the Access Key Id, the Access Key and the Region
4 - You are all set to use the CLI

Outsystems environment

1. You net to install from forge the following:
   - DynamoDB connector
   - Amazon Simple storage Service
   - Camera Plugin
   - Common Plugin
2. You need to install this custom version of Amazon Rekognition (The .NET code had to be updated to use the AWSSDK.Core updated version)
3. Install the starter version of the Restaurant Finder (AWS) app
4. Configure the site properties of the application (this keys are an example and not real):

   AWS_AccessKeyId - AKIAX6HMOSDKE3LAO39D
   AWS_Region - EUWest1
   AWS_SecretAccessKey - ShnHFsTIn8LD5Wi9sXPqksdiGD6849Lkfje73K2
   DynamboDB_Table - Restaurant
   S3_ImageBucketName - os-restaurant-images
5. Open the **RestaurantFinderAWS** module on service studio, go to the client action **GetMapKey** and in the assign put in your Google Maps Key. This google maps key needs to have the **Maps Static API** active:



GMapsKey
x.y = "AIzaSyBEyzFyAVk7KfV7tMwb5Nxy3zEe0000000"
Variable
x.y = Value

Demo

Summary

In this demo we are going to show how you can use OutSystems and AWS together.
First we are going to show the data that exists already in the DynamoDB using the AWS CLI in the command line.
Then we will open the Restaurant Finder AWS application in the service studio. We will go through the existing screen and explain how we are consuming the AWS services through a .Net extension.
We will get an item in the AWS CLI, copy the JSON and import it to a structure in Service Studio to show how easy it is to map structures and how the pre existing mapping was done. We will not use this structure since it is already done.
We are going to create a new screen for the Restaurant Detail. We are going to use the template for location detail. We are going to fetch the date using a "get-item" pre build action. We will replace the data.
We will create a server action that receives a binary, checks using the Amazon Rekognition for labels of "Restaurant" or "Food", and if they exist will send them to the S3 bucket.If not, it will give a warning.
We will create a client action that takes a picture and calls this server action. Then we will test it out taking a picture of nothing (and giving the warning that this picture is not of a restaurant or food) and another picture of food (you can take a picture of an image of food)
Then using the CLI we will see that picture in the bucket and copy it to the desktop to show.

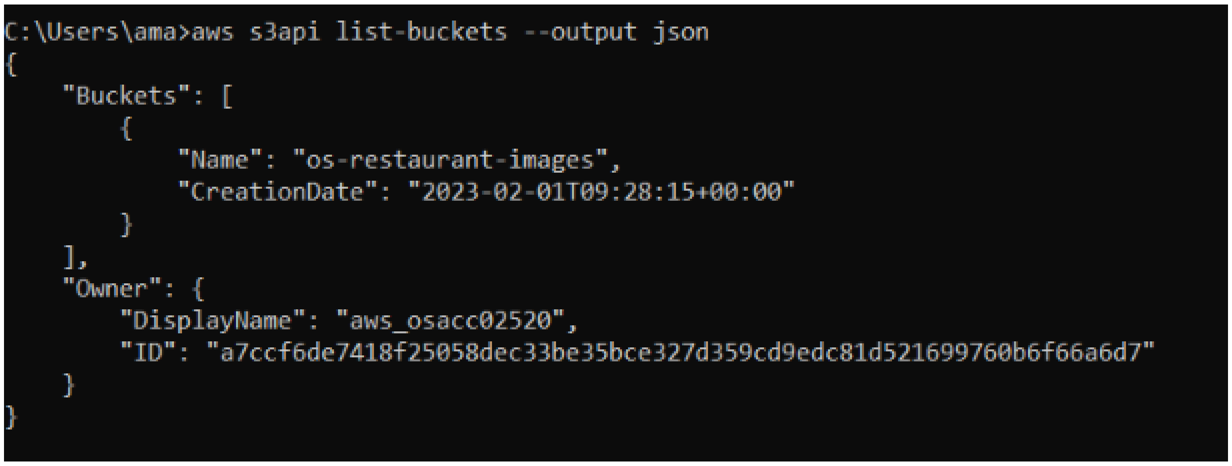
We are using OS to create a quick mobile app that integrates in the AWS Ecosystem. We are quickly showing the list of restaurants and can easily create a screen to show detail. We can easily create a feature for the user to share pictures of the restaurants. And we are pre approving those pictures (using Amazon Rekognition) and sending them to S3. At the end of the demo we can say as an example, that from the S3 bucket, a Lambda function can be triggered that processes the image, creates thumbnails and adds it to our DynamoDB database closing the cycle.

Demo Steps

These steps assume that all prerequisites are installed and configured correctly. The names being used in the commands are the ones given as examples in the site properties. All the AWS CLI commands are expressed for Windows CMD.

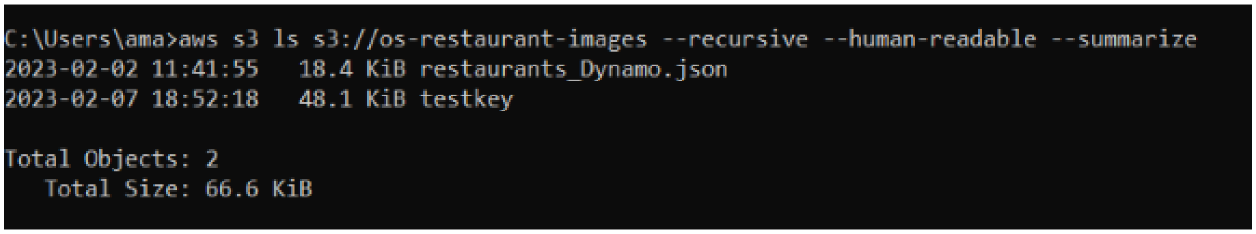
Starting with the AWS CLI

1) Open your console and show which S3 buckets exist using the command **aws s3api list-buckets --output json**



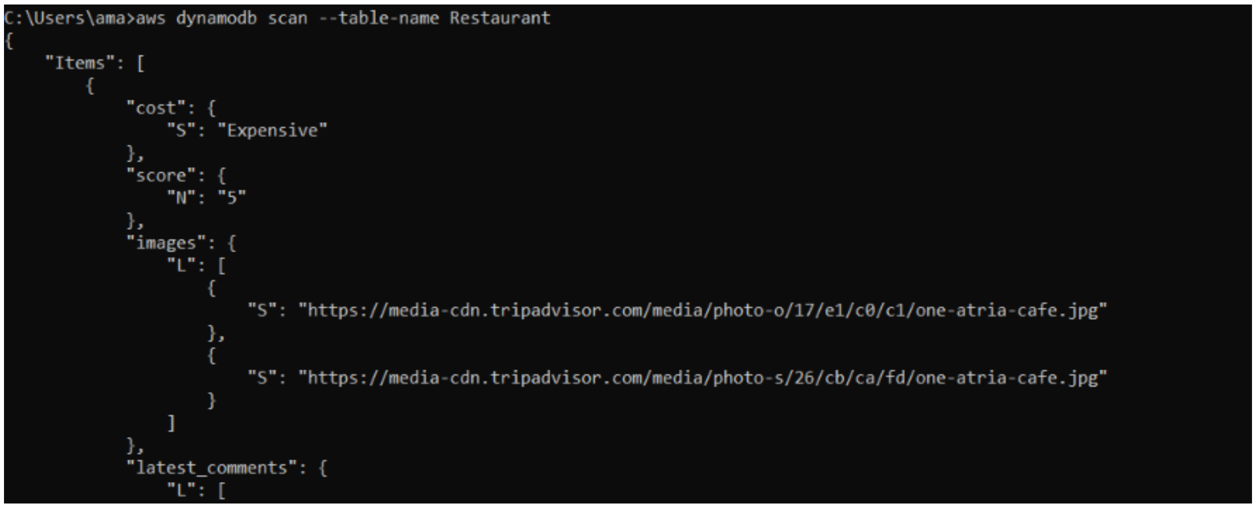
```
C:\Users\ama>aws s3api list-buckets --output json
{
    "Buckets": [
        {
            "Name": "os-restaurant-images",
            "CreationDate": "2023-02-01T09:28:15+00:00"
        }
    ],
    "Owner": {
        "DisplayName": "aws_osacc02520",
        "ID": "a7ccf6de7418f25058dec33be35bce327d359cd9edc81d521699760b6f66a6d7"
    }
}
```

2) Then list the content of your bucket using the command **aws s3 ls s3://os-restaurant-images --recursive --human-readable --summarize** . This will show that the image that you're going to upload in the future is not there.



```
C:\Users\ama>aws s3 ls s3://os-restaurant-images --recursive --human-readable --summarize
2023-02-02 11:41:55   18.4 KiB restaurants_Dynamo.json
2023-02-07 18:52:18   48.1 KiB testkey

Total Objects: 2
   Total Size: 66.6 KiB
```

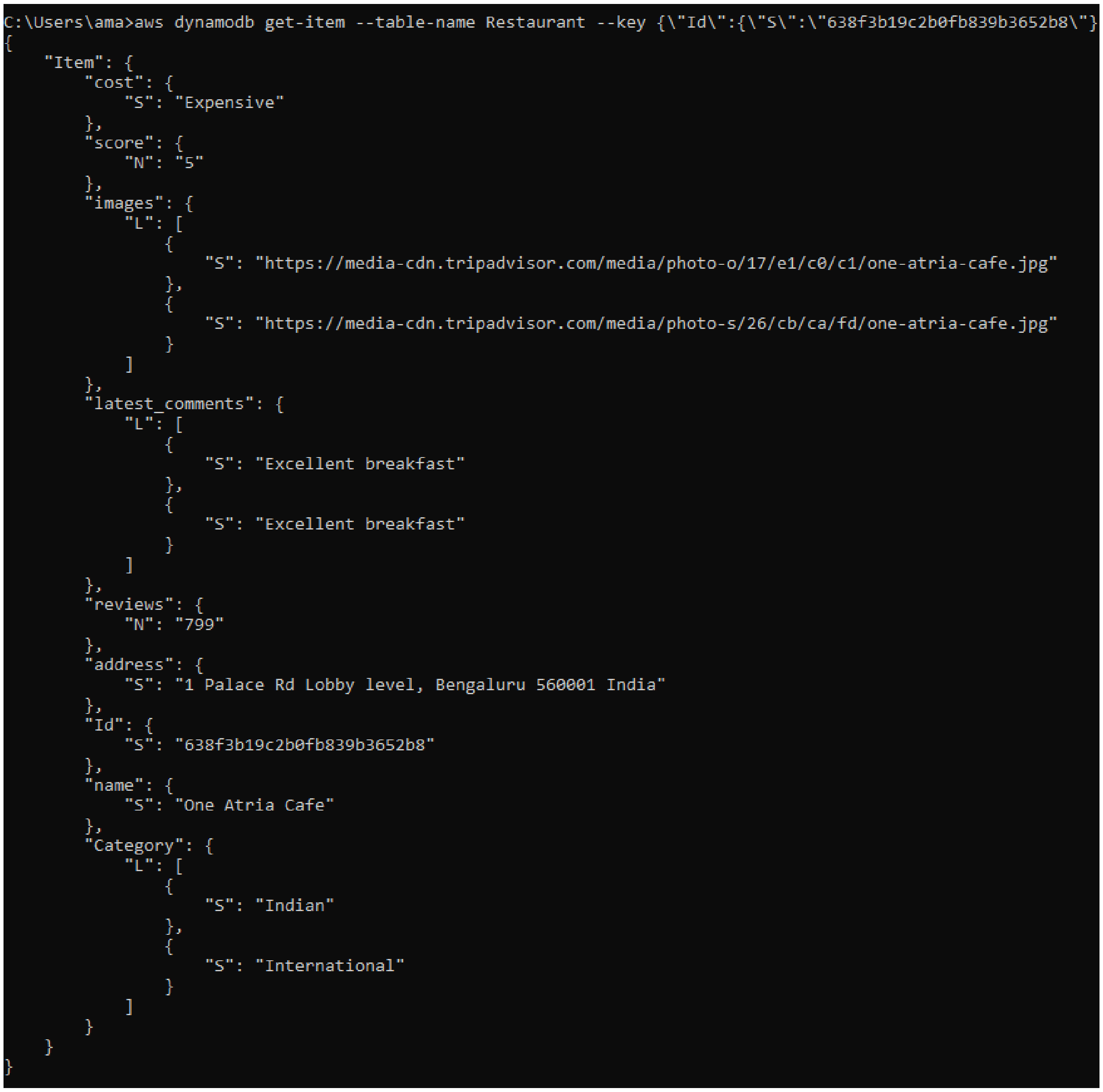
3) Open your console and show the DynamoDB Restaurant DB using the command **aws dynamodb scan --table-name Restaurant**



```
C:\Users\ama>aws dynamodb scan --table-name Restaurant
{
    "Items": [
        {
            "cost": {
                "S": "Expensive"
            },
            "score": {
                "N": "5"
            },
            "images": {
                "L": [
                    {
                        "S": "https://media-cdn.tripadvisor.com/media/photo-o/17/e1/c0/c1/one-atria-cafe.jpg"
                    },
                    {
                        "S": "https://media-cdn.tripadvisor.com/media/photo-s/26/cb/ca/fd/one-atria-cafe.jpg"
                    }
                ]
            },
            "latest_comments": {
                "L": [
```

4) You can press enter a bit to show more information. Then let’s show only one item so you can copy the JSON. Using the id from the first record do a “get-item” using the command **aws dynamodb get-item --table-name Restaurant --key**

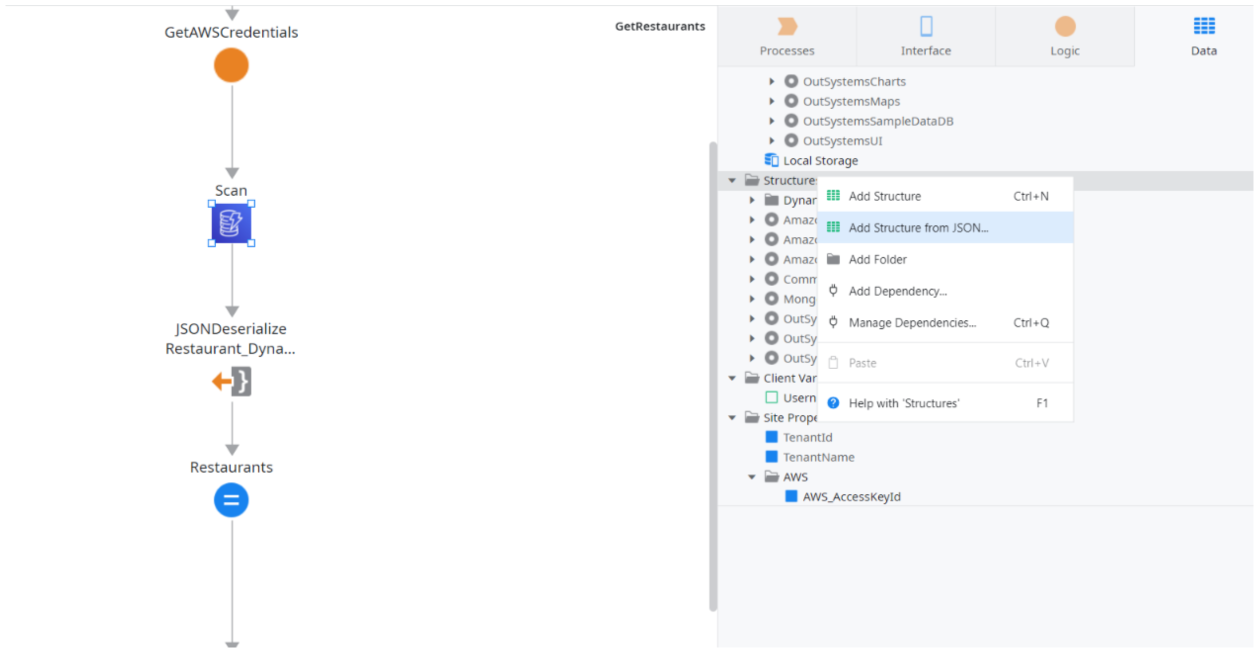
{\"Id\":{\"S\":\"638f3b19c2b0fb839b3652b8\"}}



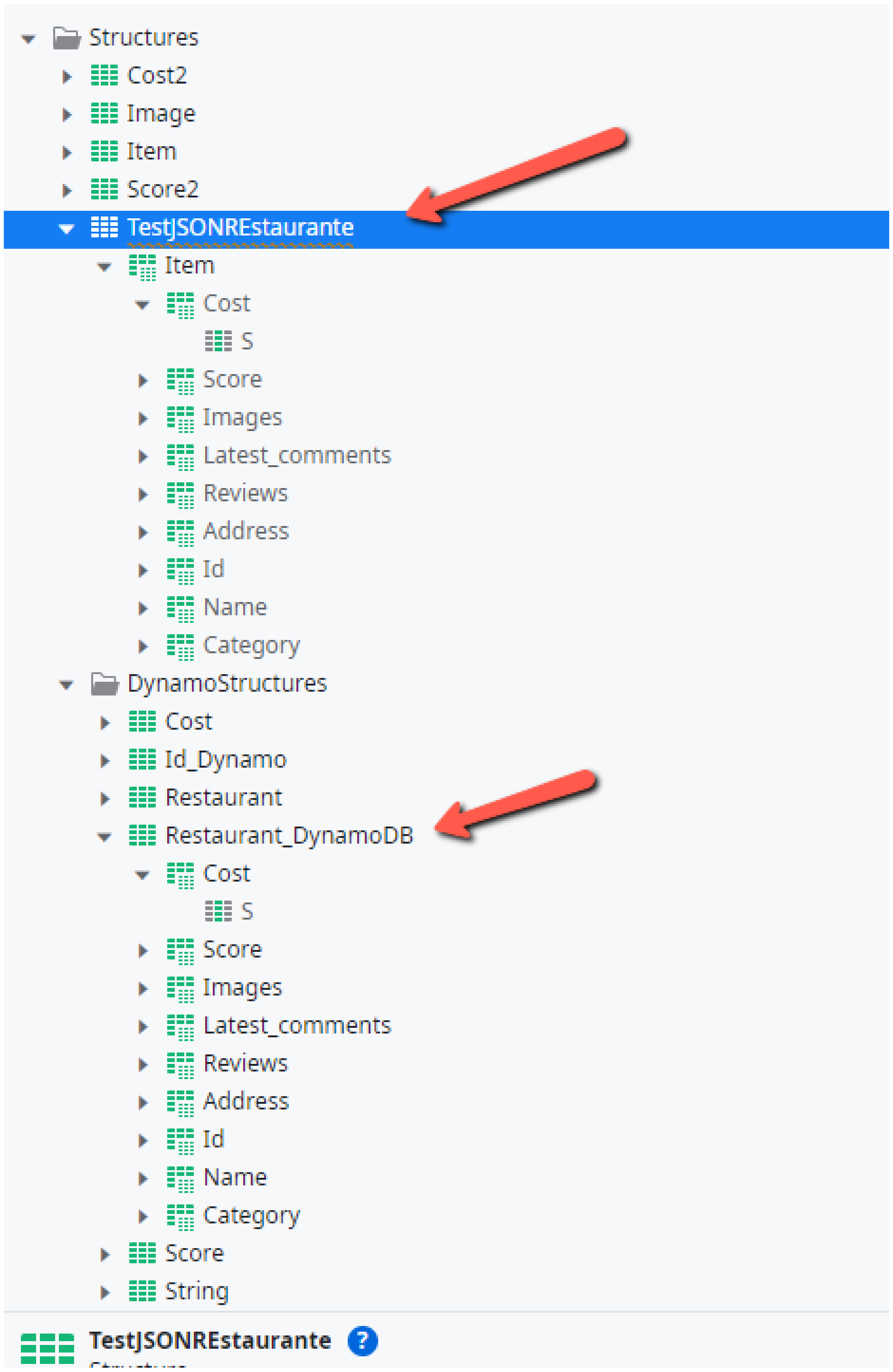
```
C:\Users\ama>aws dynamodb get-item --table-name Restaurant --key {\"Id\":{\"S\":\"638f3b19c2b0fb839b3652b8\"}}
{
    "Item": {
        "cost": {
            "S": "Expensive"
        },
        "score": {
            "N": "5"
        },
        "images": {
            "L": [
                {
                    "S": "https://media-cdn.tripadvisor.com/media/photo-o/17/e1/c0/c1/one-atria-cafe.jpg"
                },
                {
                    "S": "https://media-cdn.tripadvisor.com/media/photo-s/26/cb/ca/fd/one-atria-cafe.jpg"
                }
            ]
        },
        "latest_comments": {
            "L": [
                {
                    "S": "Excellent breakfast"
                },
                {
                    "S": "Excellent breakfast"
                }
            ]
        },
        "reviews": {
            "N": "799"
        },
        "address": {
            "S": "1 Palace Rd Lobby level, Bengaluru 560001 India"
        },
        "Id": {
            "S": "638f3b19c2b0fb839b3652b8"
        },
        "name": {
            "S": "One Atria Cafe"
        },
        "Category": {
            "L": [
                {
                    "S": "Indian"
                },
                {
                    "S": "International"
                }
            ]
        }
    }
}
```

Showing the data in Service Studio

5) Select the JSON and copy it. Go to **Service Studio** and open the **RestaurantFinderAWS** module. Drill down on the existing screen showing how the data is fetched. Show the **Scan** action, explaining that it is an extension done in .Net code. While explaining how we get the data and deserialize it show how to create a structure from JSON. Go to Data, right click in "structures" and select **Add Structure from JSON**:

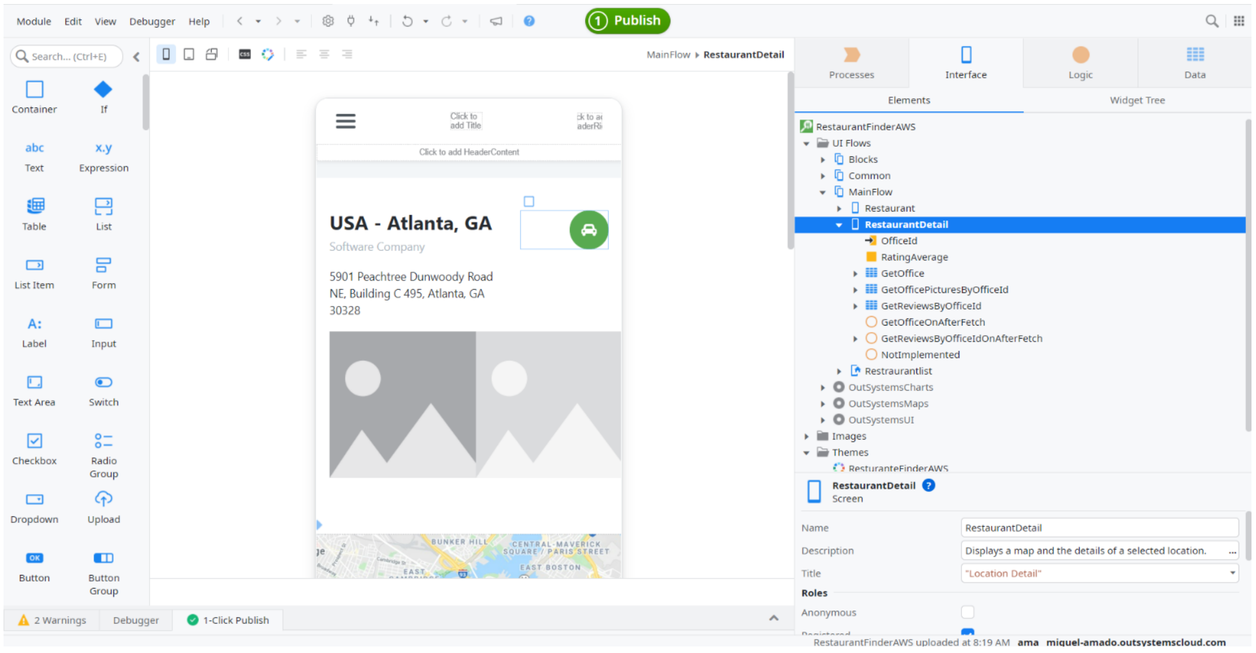


6) Paste the copied JSON and give it a name. Show how the created structures are equal to the ones already existing



Creating a detail Screen

7) Now that how we get the data is explained, let's move onto creating the new screen. Go to the **main flow** in the interface and click on the right mouse button. Add a screen. From the templates select the "**Location Detail**". Give it the name "**RestaurantDetail**" and create it.

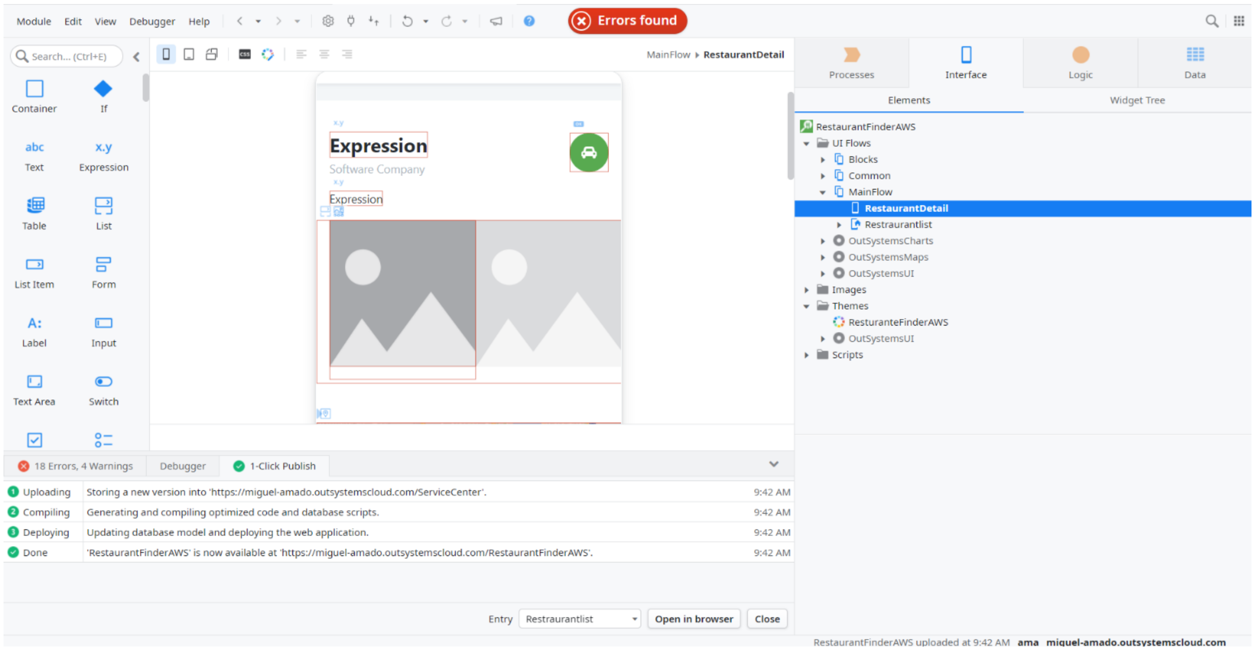


RestaurantDetail Screen

Name	RestaurantDetail
Description	Displays a map and the details of a selected location.
Title	"Location Detail"
Roles	
Anonymous	

RestaurantFinderAWS uploaded at 8:19 AM ama miguel-amado.outsystemscloud.com

8) Select variables, aggregates and client actions from that newly created screen and delete them all (since we are going to fetch our own data)

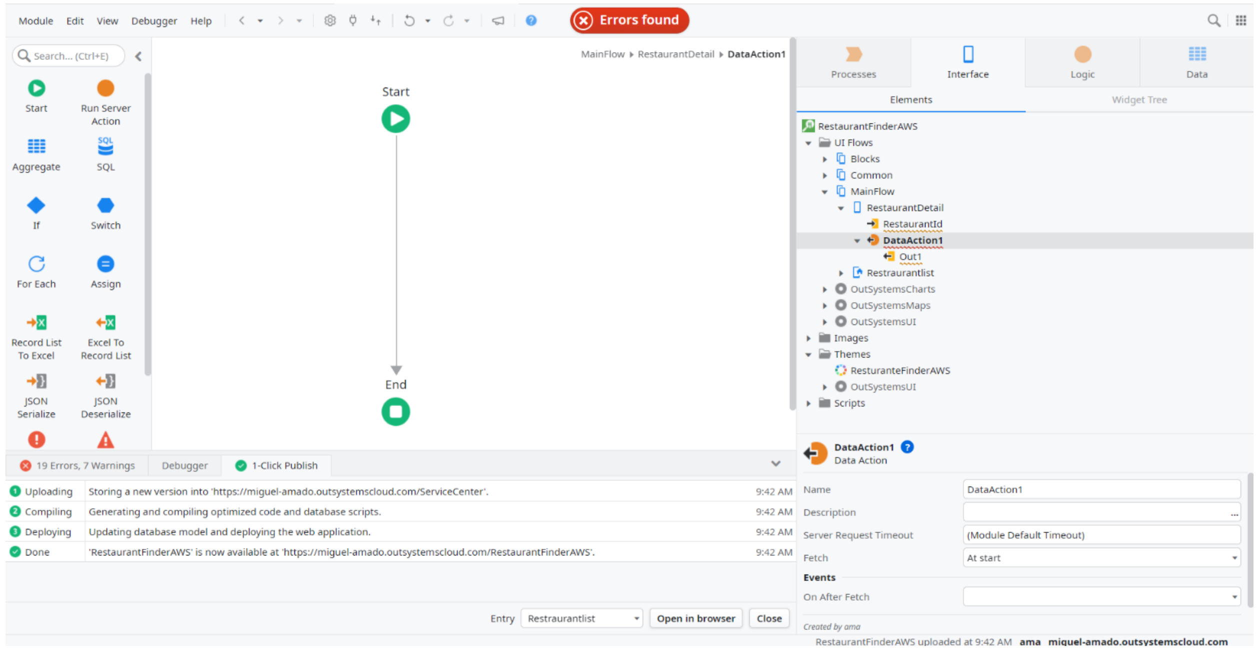


```
1 Uploading   Storing a new version into 'https://miguel-amado.outsystemscloud.com/ServiceCenter'.                      9:42 AM
2 Compiling   Generating and compiling optimized code and database scripts.                                             9:42 AM
3 Deploying   Updating database model and deploying the web application.                                                9:42 AM
  Done        'RestaurantFinderAWS' is now available at 'https://miguel-amado.outsystemscloud.com/RestaurantFinderAWS'.  9:42 AM
```

RestaurantFinderAWS uploaded at 9:42 AM ama miguel-amado.outsystemscloud.com

9) Click on the right button on top of the RestaurantDetail page and add an **input parameter** called "**RestaurantId**" of the text type. This will have the id of the restaurant

so we can query the DynamoDB to get the information. Also click on the right mouse button on top of the RestaurantDetail and add a “**Fetch Data from other Sources**”. In this server action is where we are going to use the **get-item** action



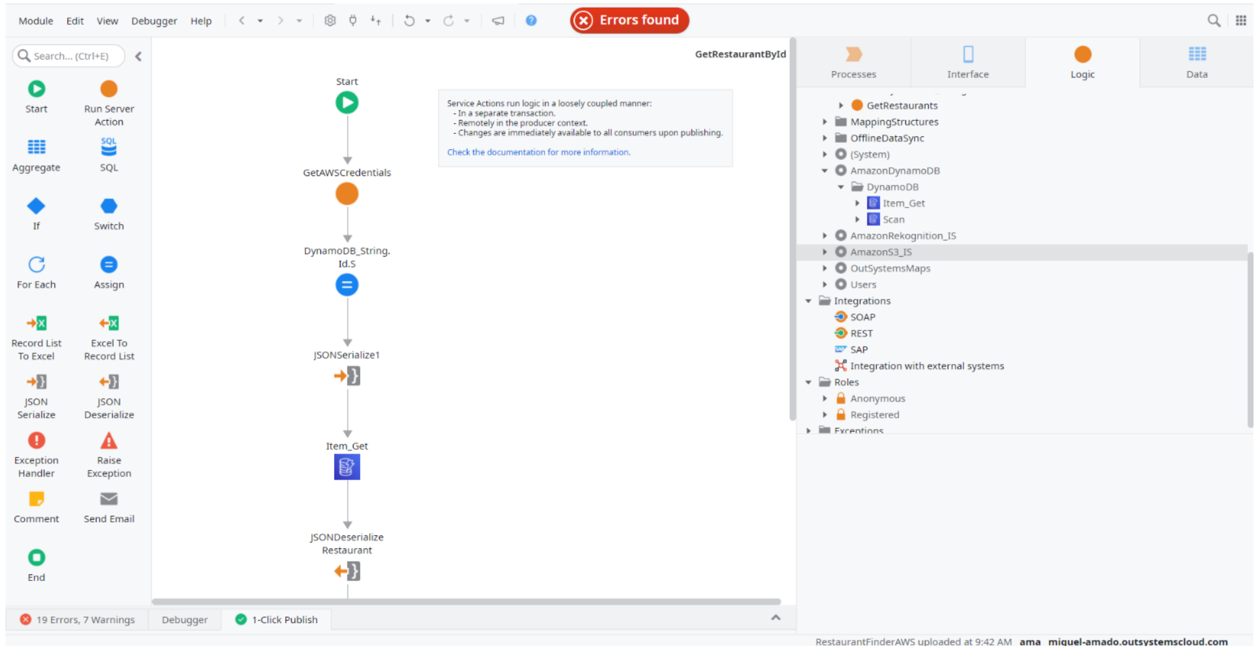
```
Uploading   Storing a new version into 'https://miguel-amado.outsystemscloud.com/ServiceCenter'.                      9:42 AM
Compiling   Generating and compiling optimized code and database scripts.                                             9:42 AM
Deploying   Updating database model and deploying the web application.                                                9:42 AM
Done        'RestaurantFinderAWS' is now available at 'https://miguel-amado.outsystemscloud.com/RestaurantFinderAWS'.  9:42 AM
```

DataAction1
Data Action

Name	DataAction1
Description	
Server Request Timeout	(Module Default Timeout)
Fetch	At start
Events	
On After Fetch	

Created by ama
RestaurantFinderAWS uploaded at 9:42 AM ama miguel-amado.outsystemscloud.com

10) Open the Logic tab and check the **GetRestaurantById** action with the **Item_Get** extension. **Show** how the query parameter “Id” is serialized and sent to the extension.



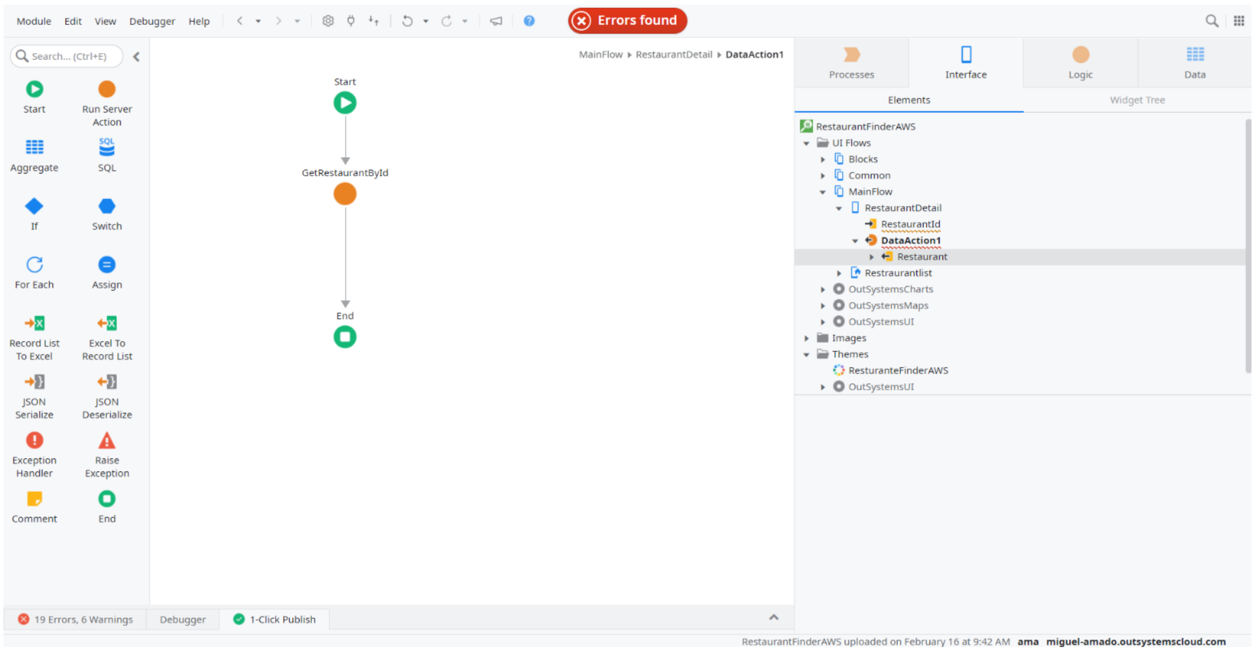
Service Actions run logic in a loosely coupled manner:
- In a separate transaction.
- Remotely in the producer context.
- Changes are immediately available to all consumers upon publishing.

Check the documentation for more information.

RestaurantFinderAWS uploaded at 9:42 AM ama miguel-amado.outsystemscloud.com

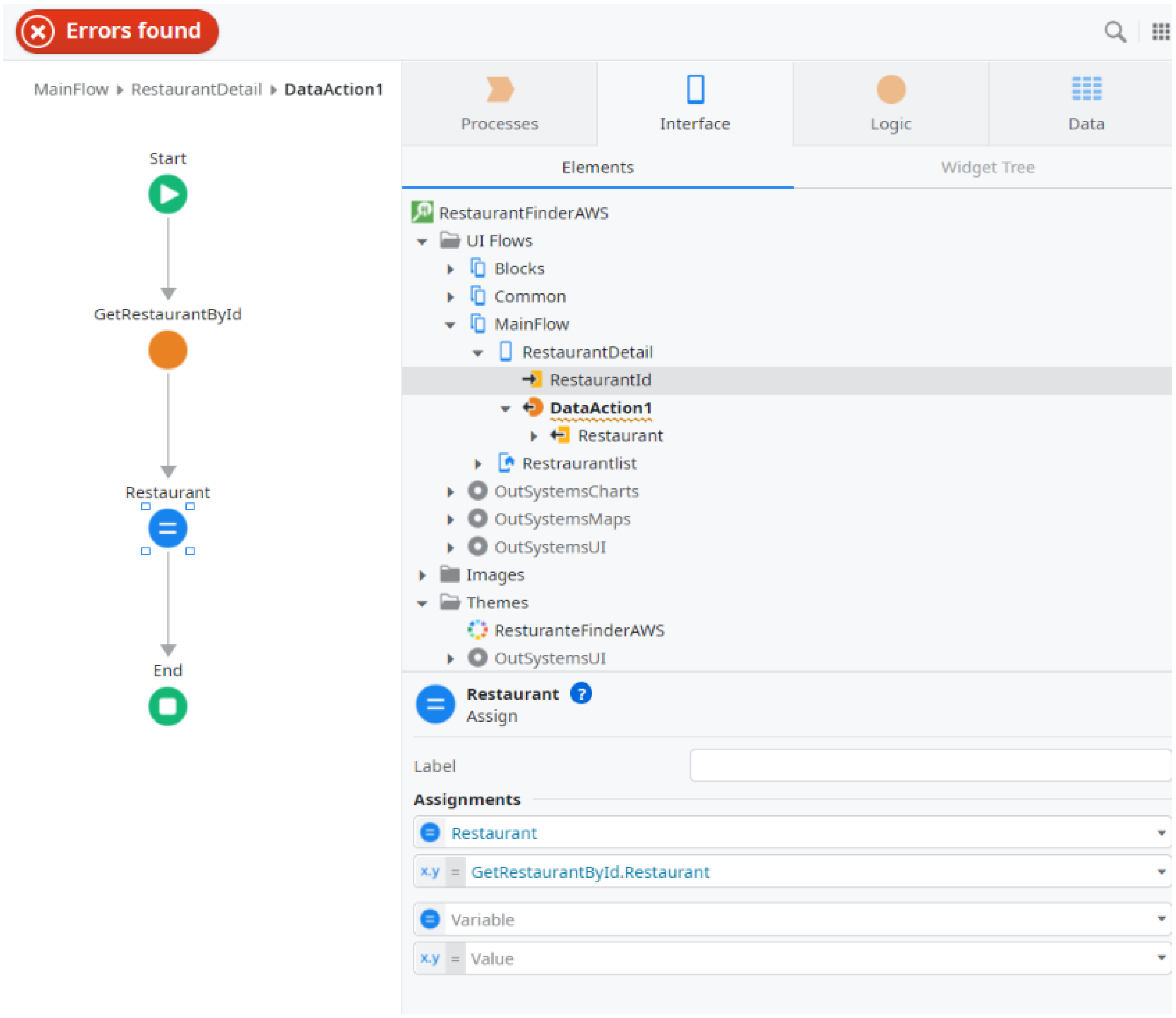
11) Drag the **GetRestaurantById** to the **DataAction1** of the **RestaurantDetail** screen. Pass the RestaurantId input to the input attribute of the **GetRestaurantById** action. Now copy the output of the first action (**GetRestaurantById**) to the **DataAction1** and delete the

“Out1” output



Module Edit View Debugger Help
Errors found
Search... (Ctrl+E)
Start
Run Server Action
Aggregate
SQL
If
Switch
For Each
Assign
Record List To Excel
Excel To Record List
JSON Serialize
JSON Deserialize
Exception Handler
Raise Exception
Comment
End
MainFlow ▸ RestaurantDetail ▸ DataAction1
Start
GetRestaurantById
End
Processes
Interface
Logic
Data
Elements
Widget Tree
RestaurantFinderAWS
UI Flows
Blocks
Common
MainFlow
RestaurantDetail
RestaurantId
DataAction1
Restaurant
Restraurantlist
OutSystemsCharts
OutSystemsMaps
OutSystemsUI
Images
Themes
ResturanteFinderAWS
OutSystemsUI
19 Errors, 6 Warnings
Debugger
1-Click Publish
RestaurantFinderAWS uploaded on February 16 at 9:42 AM ama miguel-amado.outsystemscloud.com

12) Drag an assign and assign the **output** of the **GetRestaurantById** action to the **Restaurant** output structure

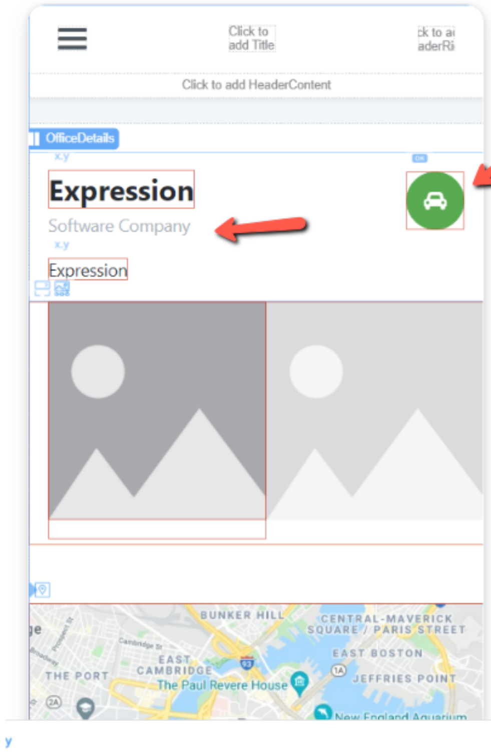


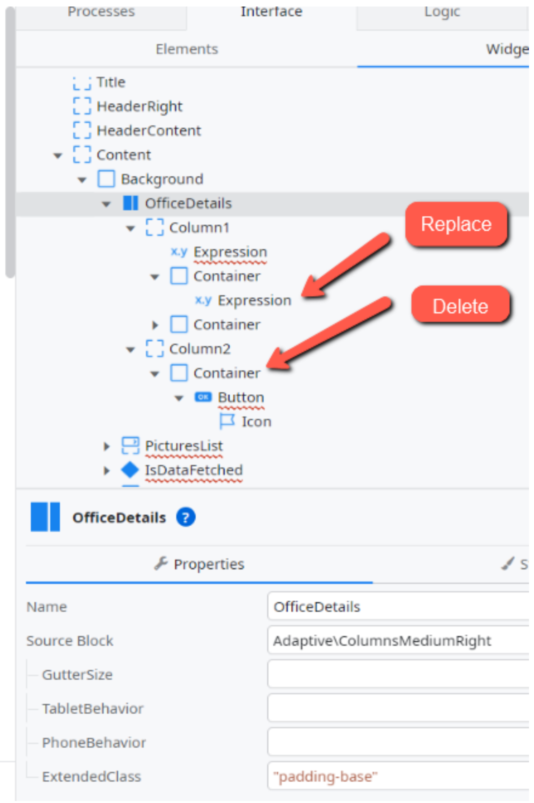
Restaurant = GetRestaurantById.Restaurant

Variable = Value

13) Now start replacing the different components of the UI with the outcomes from the DataAction1. Start by deleting the button with the car and replacing the "**Software**

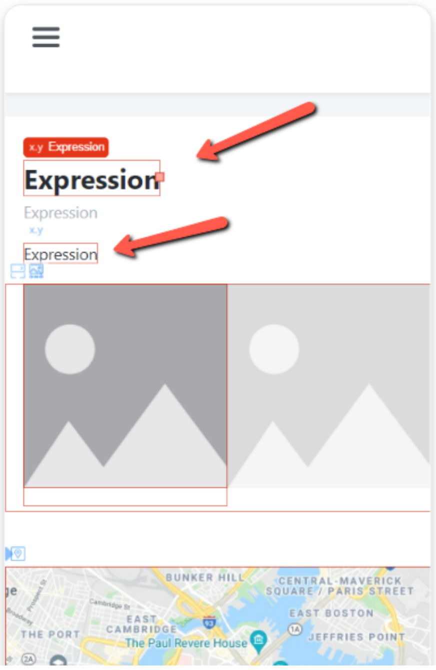
Company” with the Restaurant type (**DataAction1.Restaurant.Categories.Current**)

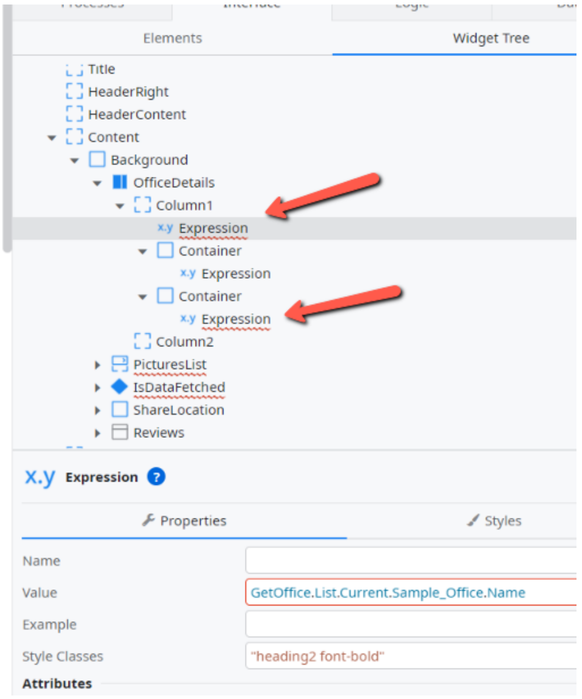




Name	OfficeDetails
Source Block	Adaptive\ColumnsMediumRight
GutterSize	
TabletBehavior	
PhoneBehavior	
ExtendedClass	"padding-base"

14) Replace the Title for the “**DataAction1.Restaurant.Name**” and the Address for the “**DataAction1.Restaurant.Address**”

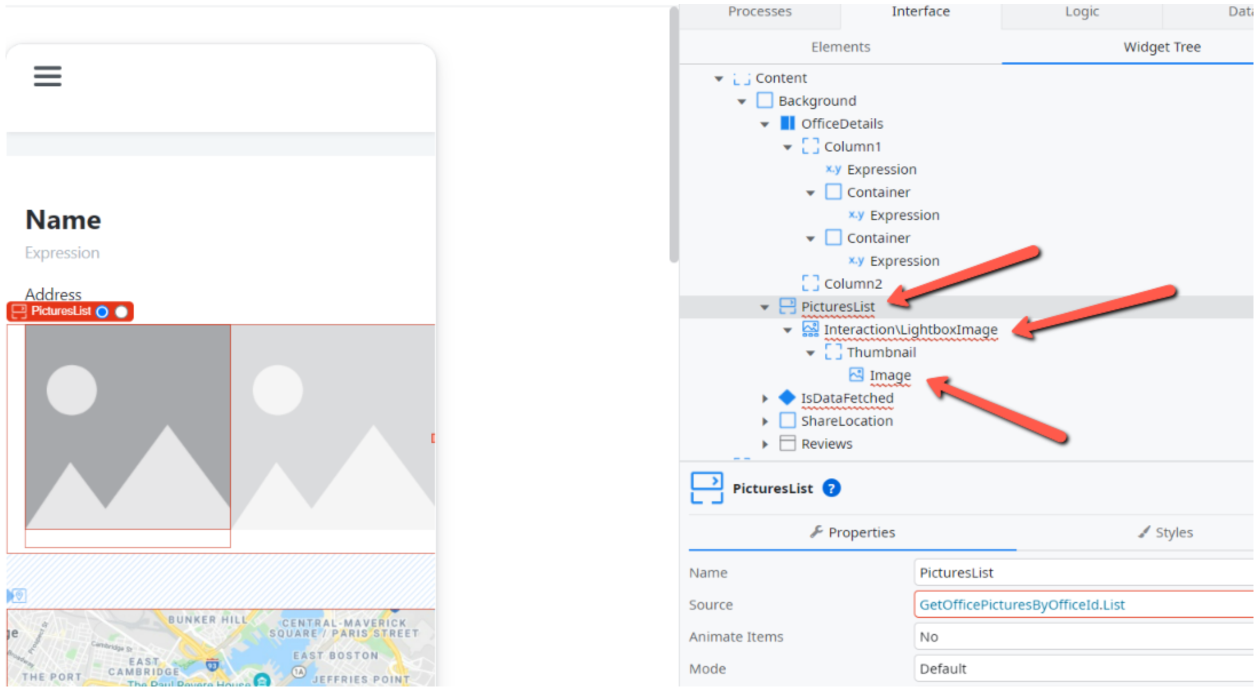




Name	
Value	GetOffice.List.Current.Sample_Office.Name
Example	
Style Classes	"heading2 font-bold"

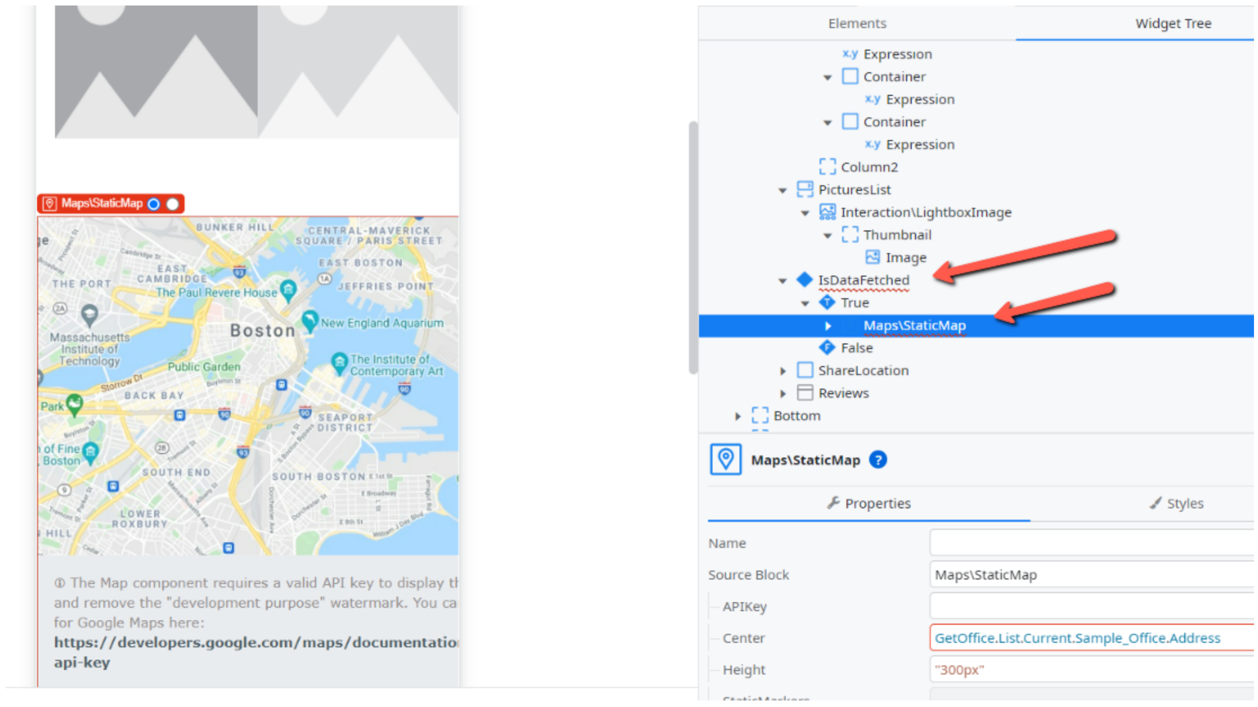
15) In the “**PictureList**” Replace the Source for the “**DataAction1.Restaurant.Images**”, the **LightboxImage** replace the **Group** attribute with the “**DataAction1.Restaurant.Name**”

and in the **Image** of the Thumbnail, change the Type to “**External URL**” and the URL to “**DataAction1.Restaurant.Images.Current**”



Name	PicturesList
Source	GetOfficePicturesByOfficeId.List
Animate Items	No
Mode	Default

16) Now let’s take care of the **staticmap**. Go to the “**IsDataFetched**” if statement and change the condition to “**DataAction1.IsDataFetched**”. Then on the StaticMap, for the APIKey use the already built function “**GetMapKey()**” and for the “**Center**” select the “**DataAction1.Restaurant.Address**”. Same for the StaticMarkers - [0] - **Location**

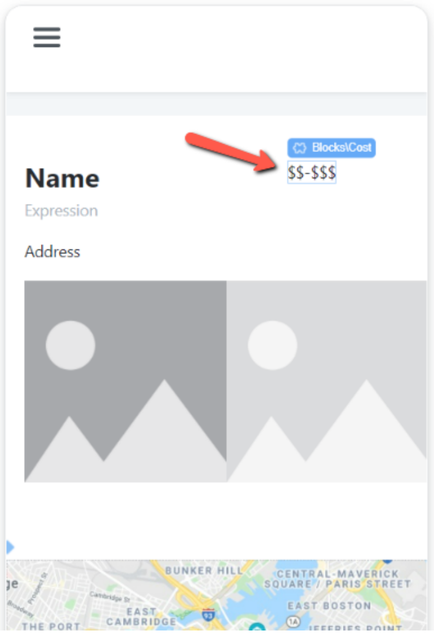


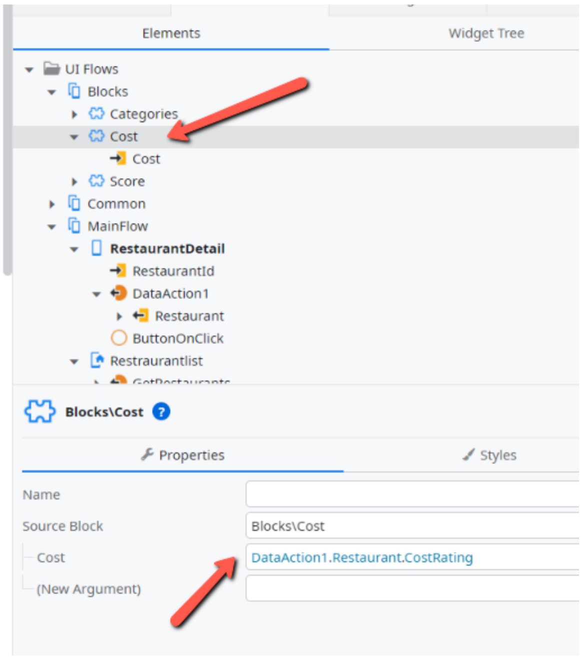
ⓘ The Map component requires a valid API key to display t
and remove the "development purpose" watermark. You ca
for Google Maps here:
https://developers.google.com/maps/documentatio
api-key

Name	
Source Block	Maps\StaticMap
APIKey	
Center	GetOffice.List.Current.Sample_Office.Address
Height	"300px"

17) Now let’s add on the top right a block to show the price range of the Restaurant. Open the **Interface tab**.On the UI Flows you have a **Blocks flow**. Open it and you’ll see a

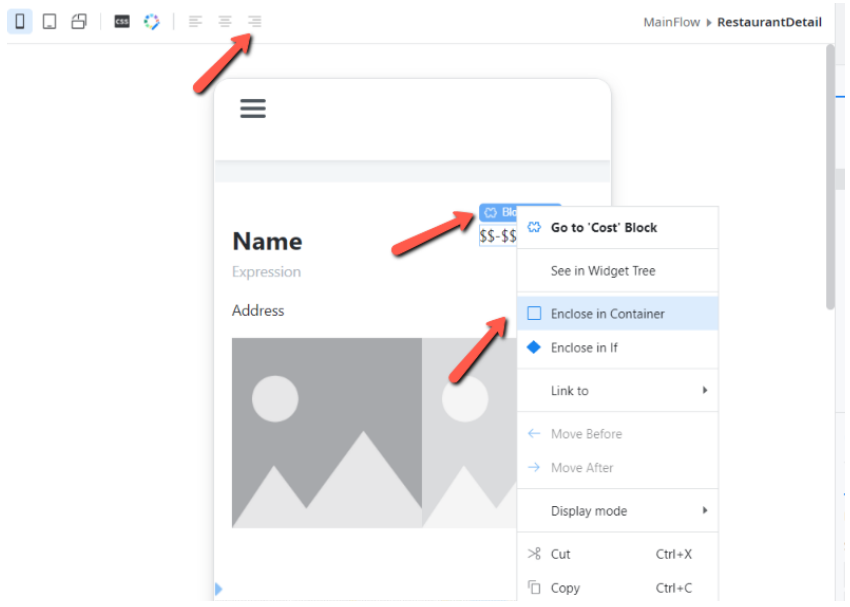
block called “**Cost**”. This is a block created by a colleague that translates the word that comes on the “**CostRating**” attribute from DynamoDB into dollar signs. We are going to reuse it on our screen. Drag the block to the right side of the Restaurant Name. On the Cost attribute of the block select “**DataAction1.Restaurant.CostRating**”





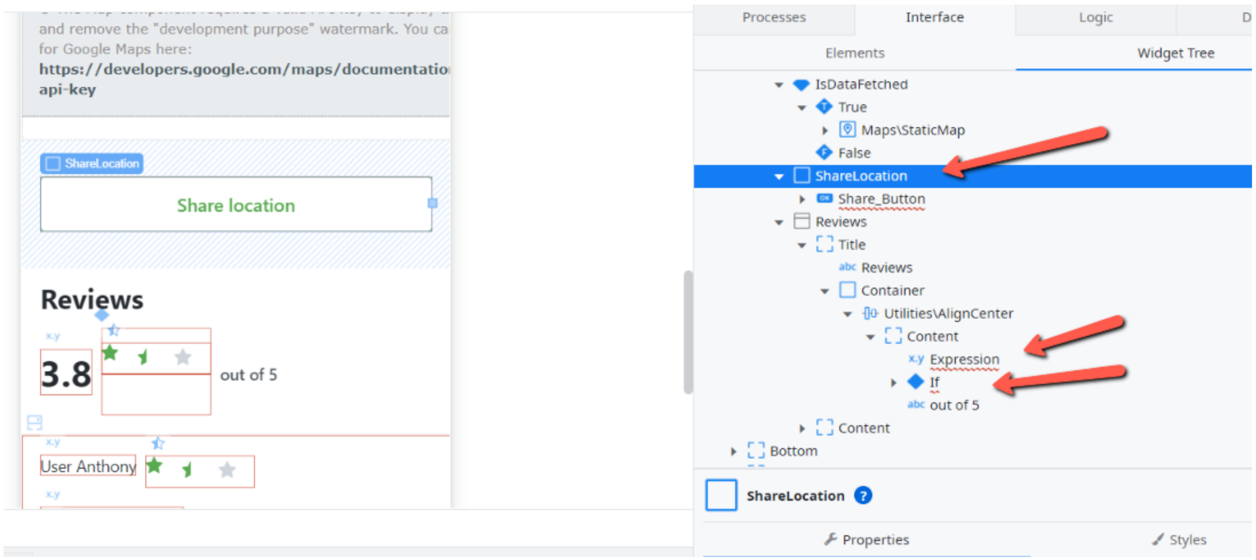
Name	
Source Block	Blocks\Cost
Cost	DataAction1.Restaurant.CostRating
(New Argument)	

18) To align the component in the page you can right click on top of the block and select the “**Enclose in Container**” option. This will create a container enclosing the block. Afterwards, on the top left of the Service Studio you have the text alignment options. Choose the align right option

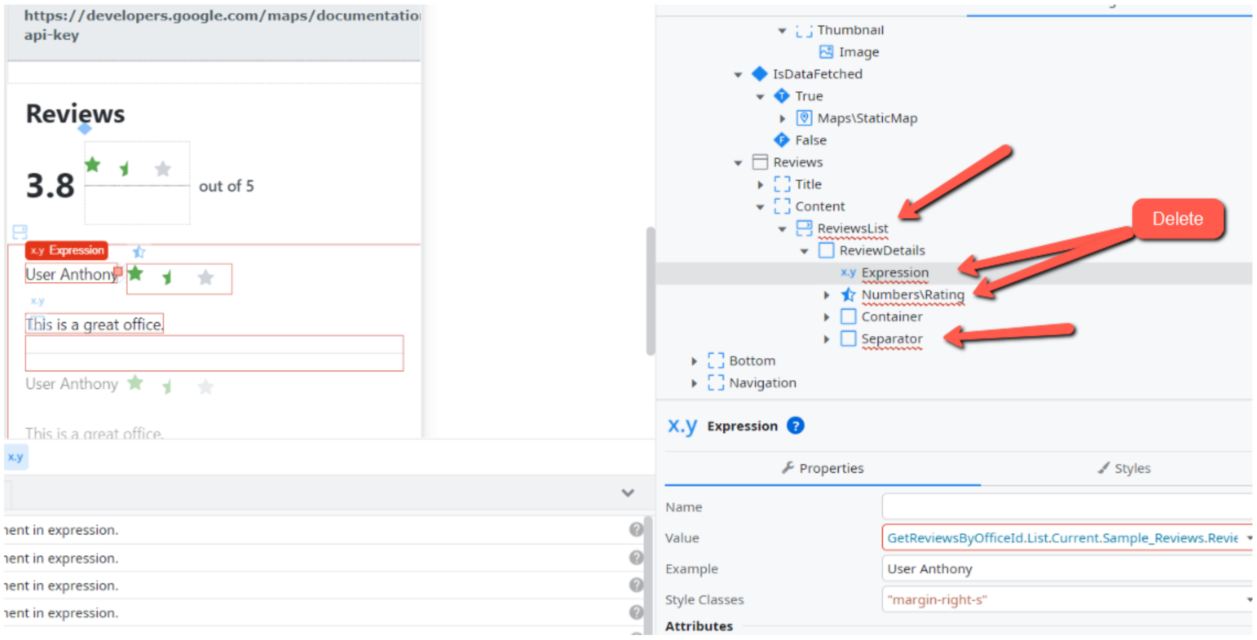


19) Continuing, Select the “**ShareLocation**” **div** and delete it. For the Reviews part, go to the first **Expression** (the big 3.8) and replace its value for the

“**DataAction1.Restaurant.Score**”. Open the next “**If**”, change it’s condition to be “**DataAction1.IsDataFetched**”, and change the Rating´s “**RatingValue**” attribute to “**DataAction1.Restaurant.Score**”



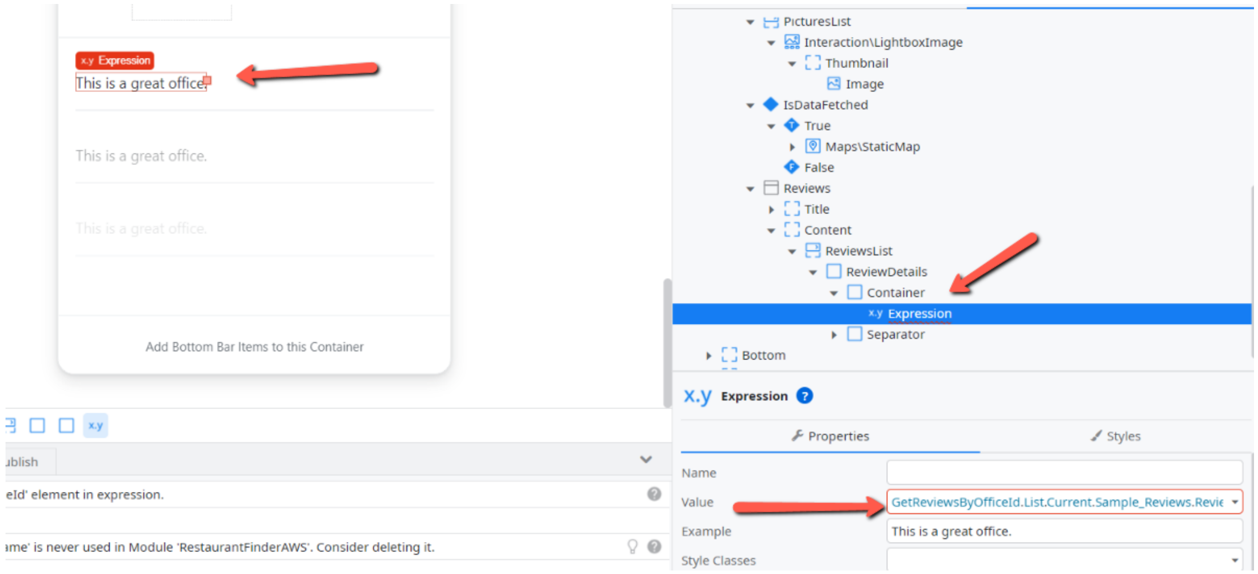
20) Now let’s go to the ReviewList. Change the list’s source to “**DataAction1.Restaurant.LatestComments**”. Delete the first expression and the rating (since for the review we only have the latest comments). Change the “Separator” div visible attribute from “GetReviewsByOfficeId.List.CurrentRowNumber <> GetReviewsByOfficeId.List.Length - 1” to “DataAction1.Restaurant.LatestComments.CurrentRowNumber <> DataAction1.Restaurant.LatestComments.Length - 1”. Here we are putting the logic of only having a separator if there is another comment following the current.



ent in expression.
ent in expression.
ent in expression.
ent in expression.

Name	
Value	GetReviewsByOfficeId.List.Current.Sample_Reviews.Revie
Example	User Anthony
Style Classes	"margin-right-s"
Attributes	

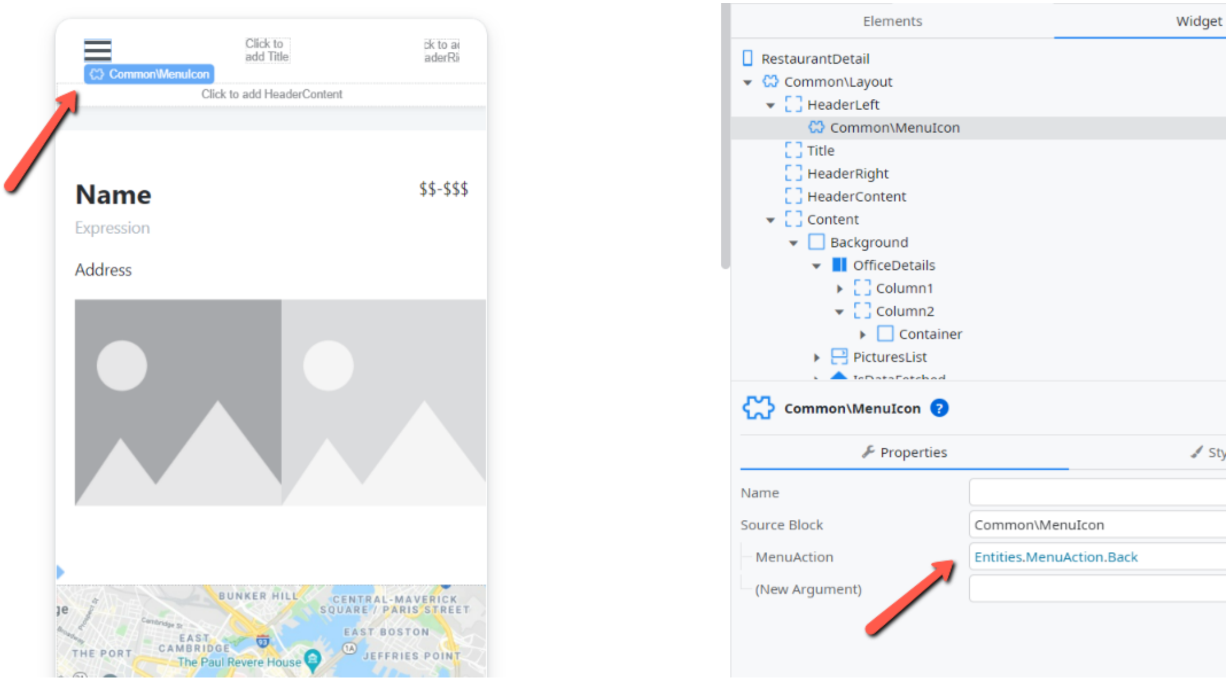
21) Finally, change the Expression on the Container before the separator for the “**DataAction1.Restaurant.LatestComments.Current**”



ıblish
eId' element in expression.
ıme' is never used in Module 'RestaurantFinderAWS'. Consider deleting it.

Name
Value GetReviewsByOfficeId.List.Current.Sample_Reviews.Revie
Example This is a great office.
Style Classes

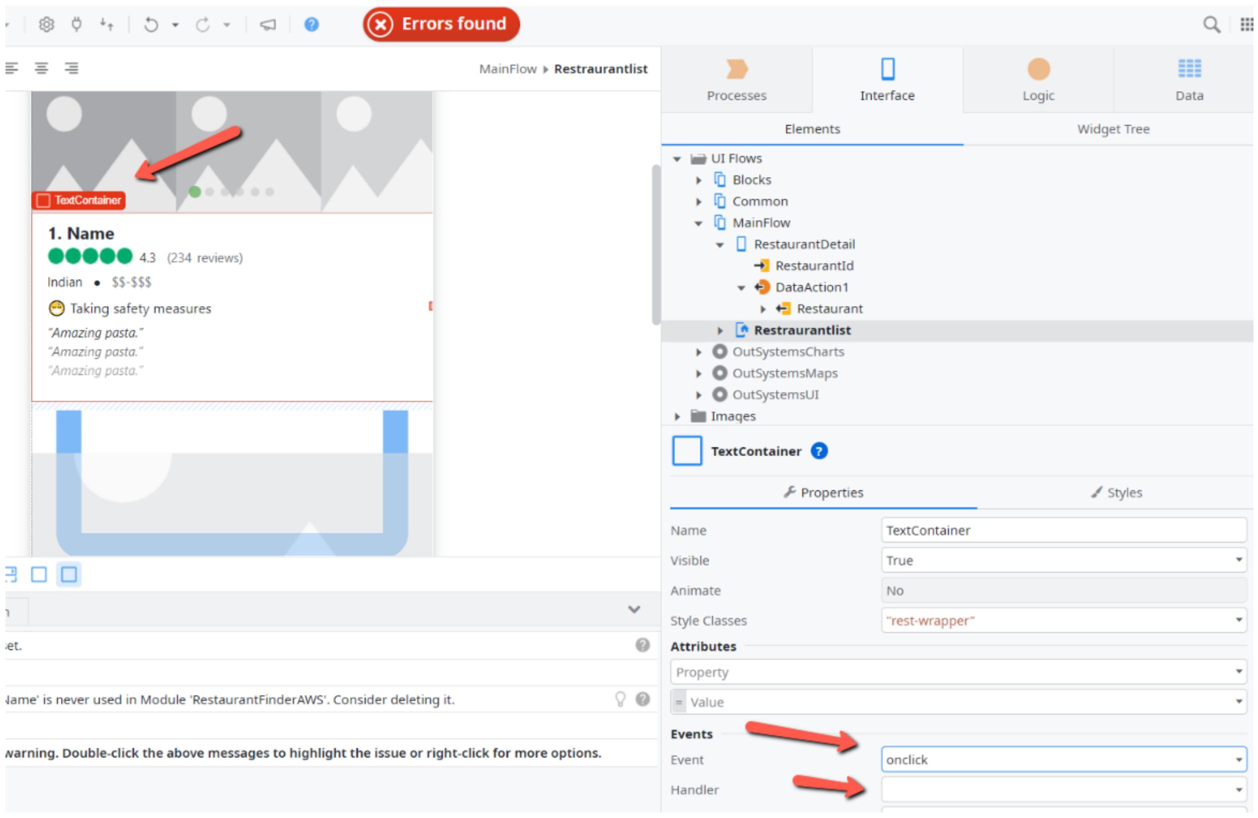
22) To finish this screen, go to the top left corner of the screen and select the menu widget. On the “**MenuAction**” attribute select “**Entities.MenuAction.Back**”.This way, the widget will have the “back” functionality



Name
Source Block Common\MenuIcon
MenuAction Entities.MenuAction.Back
(New Argument)

23) Now you should have the environment free of errors and ready to publish. We just need to create a connection between the Landing **Restaurantlist** page to this detail. Open the **Restaurantlist** page, select the **TextContainer**, and on the Events option select the

“onclick” event and on the Handler choose on the Suggestions “New Client Action”.



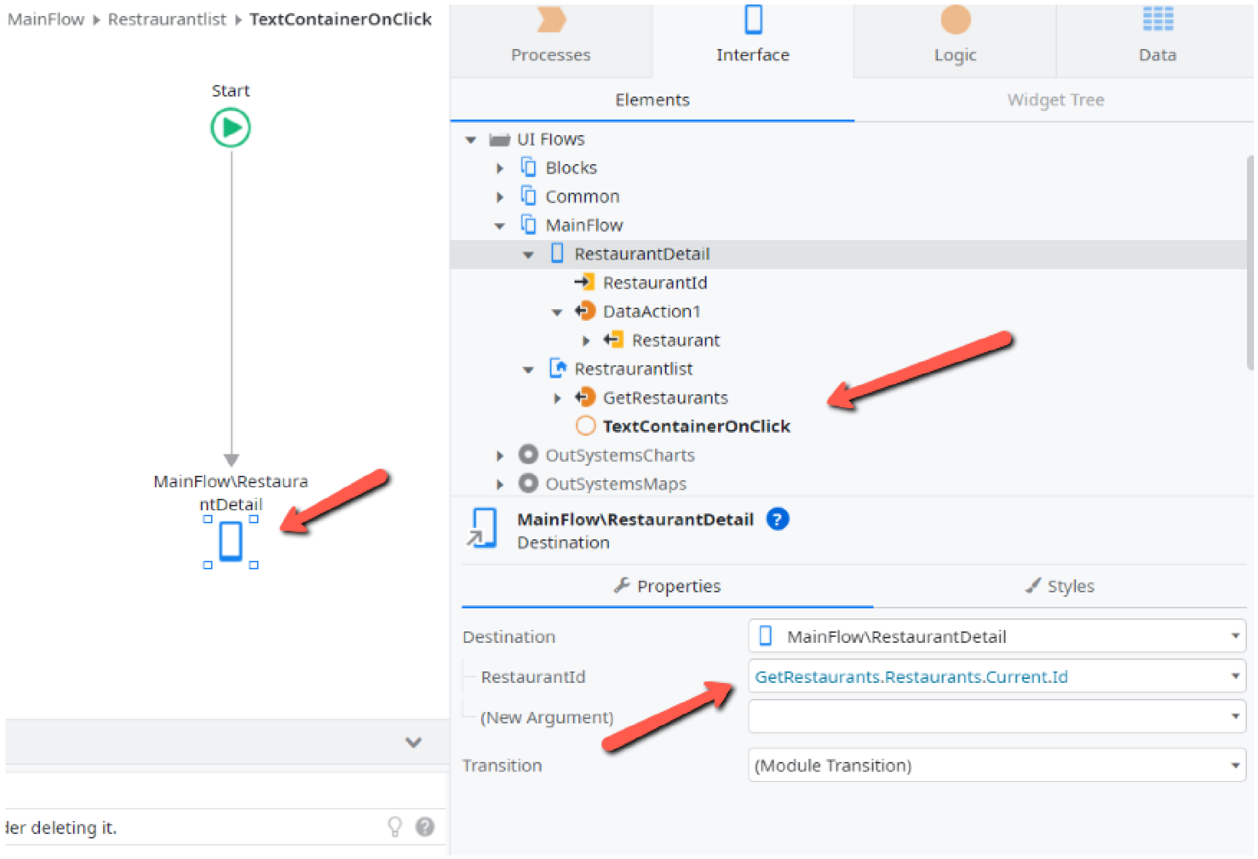
et.

lame' is never used in Module 'RestaurantFinderAWS'. Consider deleting it.

warning. Double-click the above messages to highlight the issue or right-click for more options.

Name	TextContainer
Visible	True
Animate	No
Style Classes	"rest-wrapper"
Attributes	
Property	Value
Events	
Event	onclick
Handler	

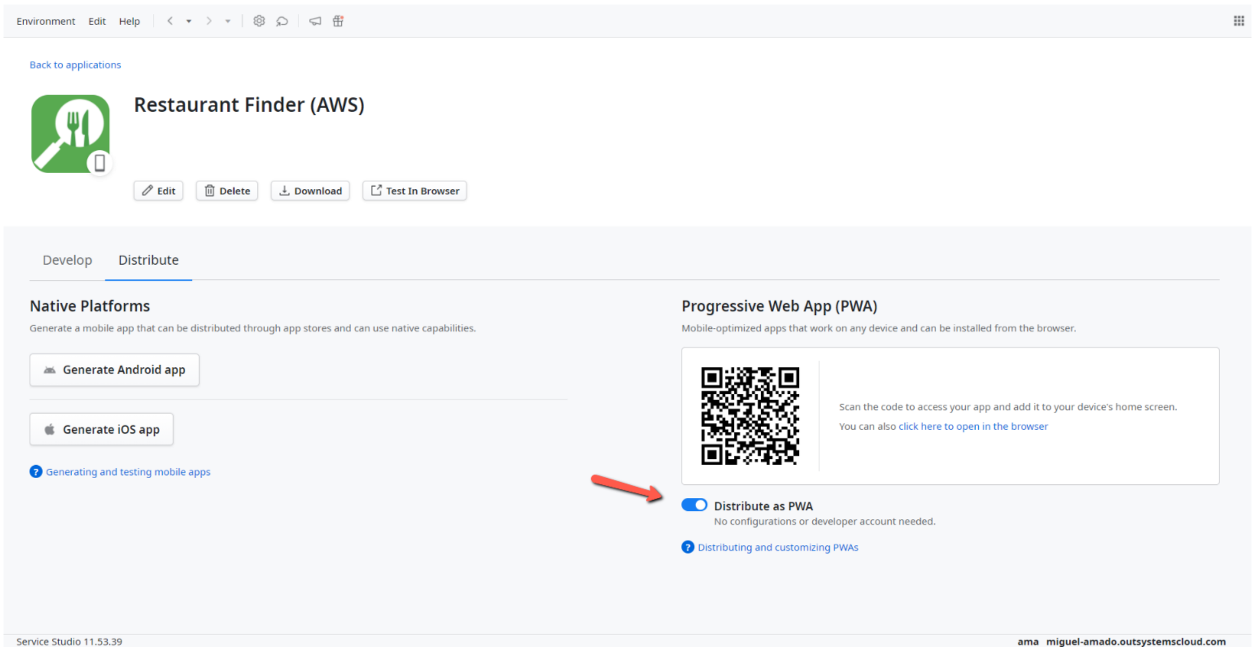
24) On the new created client action drag the **RestaurantDetail** screen to replace the end node and for the RestaurantId define the "**GetRestaurants.Restaurants.Current.Id**"



Destination	MainFlow\RestaurantDetail
RestaurantId	GetRestaurants.Restaurants.Current.Id
(New Argument)	
Transition	(Module Transition)

25) Publish and watch the results. You can turn the app into a PWA application for people to scan the qr code with their phone and try out. You can go to Service Studio, find the Restaurant Finder (AWS) application and on the Distribute tab, click on the toggle for

“Distribute as PWA” application



Restaurant Finder (AWS)

Native Platforms

Generate a mobile app that can be distributed through app stores and can use native capabilities.

Progressive Web App (PWA)

Mobile-optimized apps that work on any device and can be installed from the browser.

Scan the code to access your app and add it to your device's home screen.

You can also click here to open in the browser

Distribute as PWA

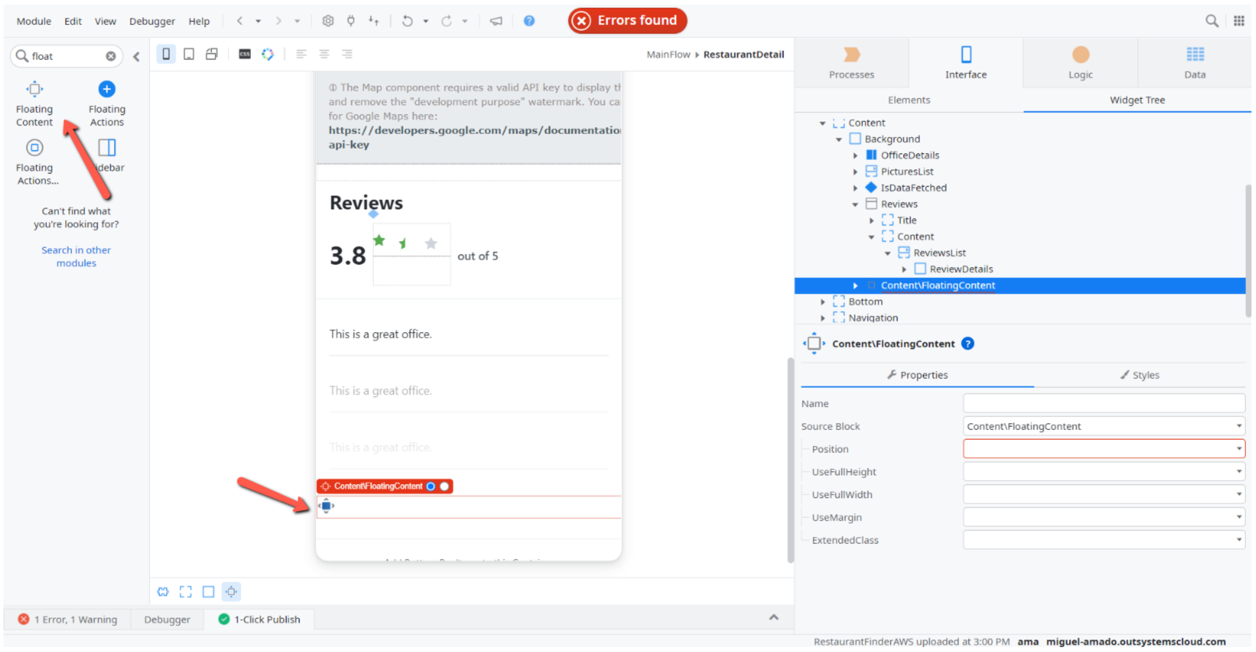
No configurations or developer account needed.

Service Studio 11.53.39

ama miguel-amado.outsystemscloud.com

Adding UI to take pictures

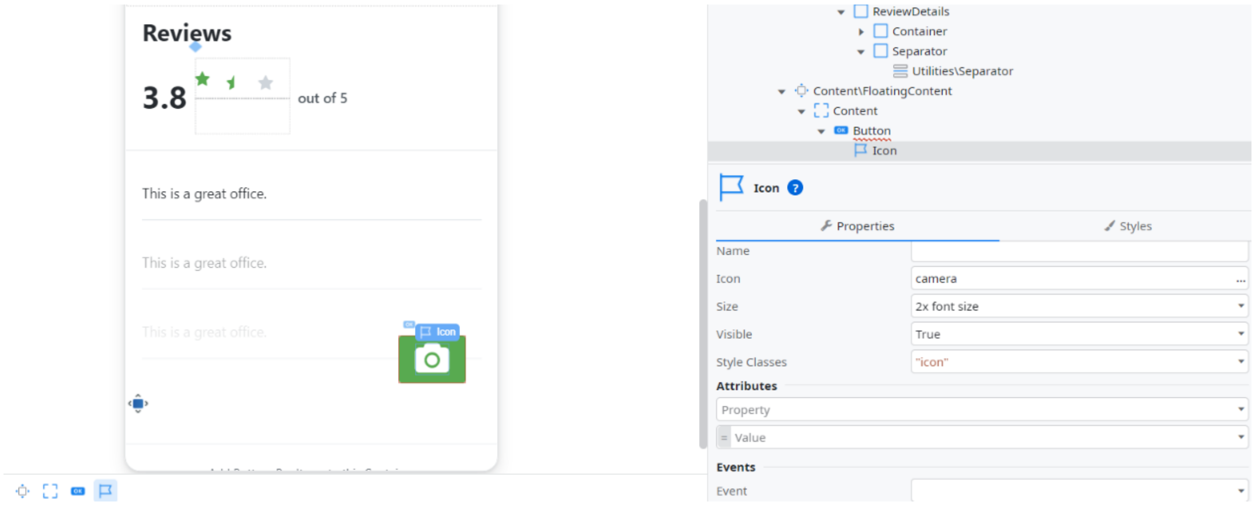
26) Let’s add an action to take pictures and send them to S3. Go to the **RestaurantDetail** Screen and add a “**Floating Content**” widget to the bottom of the page



The Map component requires a valid API key to display the map and remove the "development purpose" watermark. You can get one for Google Maps here: https://developers.google.com/maps/documentation/... api-key

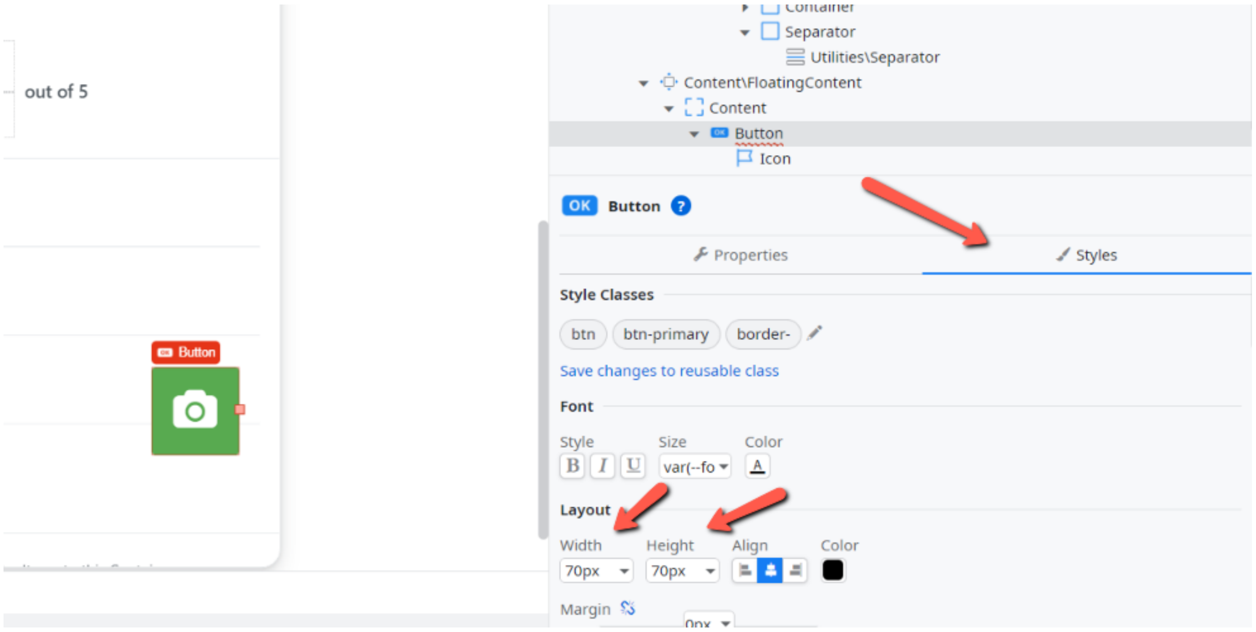
Name	
Source Block	Content\FloatingContent
Position	
UseFullHeight	
UseFullWidth	
UseMargin	
ExtendedClass	

27) Define the Position to be on the bottom right (**Entities.Position.BottomRight**). Then drag a button to the content of the “**FloatingContent**”. Replace the expression of the button for an icon widget with the camera Icon.



Name	
Icon	camera
Size	2x font size
Visible	True
Style Classes	"icon"

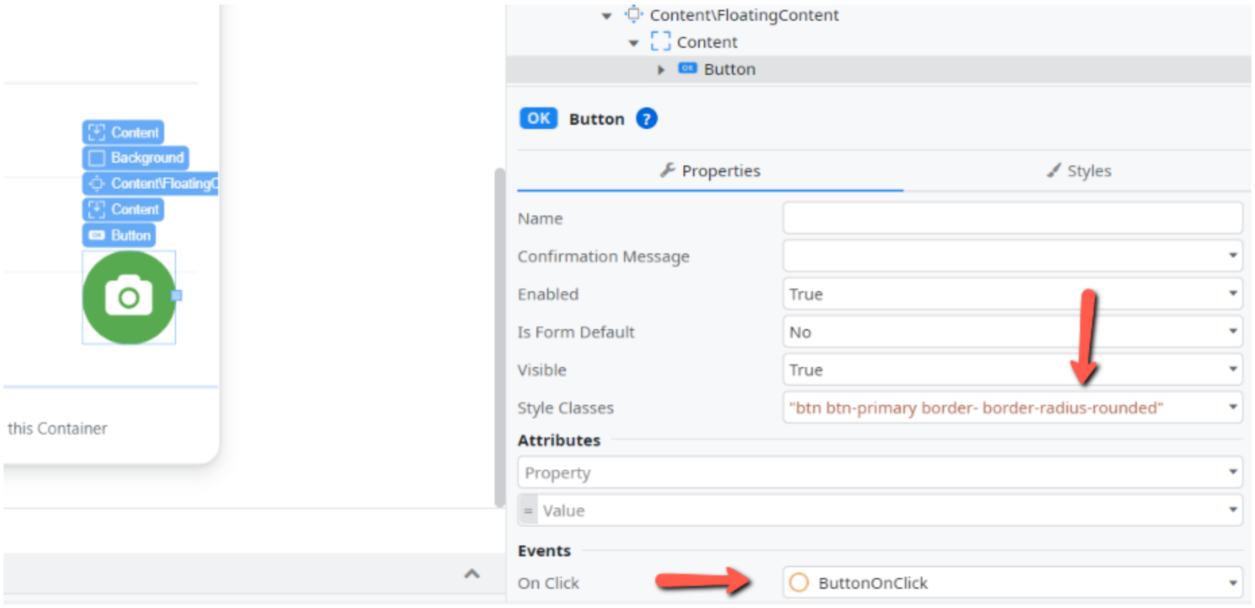
28) Select the button, change from the "**Properties**" tab to the "**Styles**" tab on the bottom right, select the width to be 70px and the height to be 70px


Style Classes: btn, btn-primary, border-

Save changes to reusable class

Layout: Width 70px, Height 70px

29) Switch back to the "**Properties**" Tab, and on the "**Style Classes**" at the class " **border-radius-rounded**". On the Event's "**onClick**", select to create a "New Client Action"


Name:

Confirmation Message:

Enabled: True

Is Form Default: No

Visible: True

Style Classes: "btn btn-primary border- border-radius-rounded"

Attributes

Property

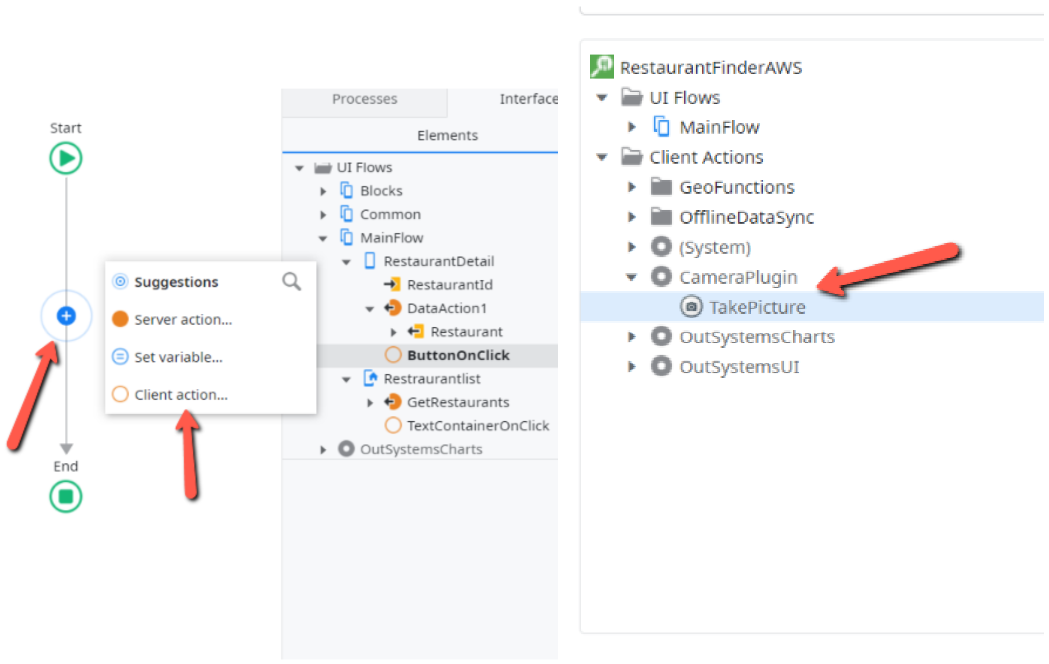
Value

Events

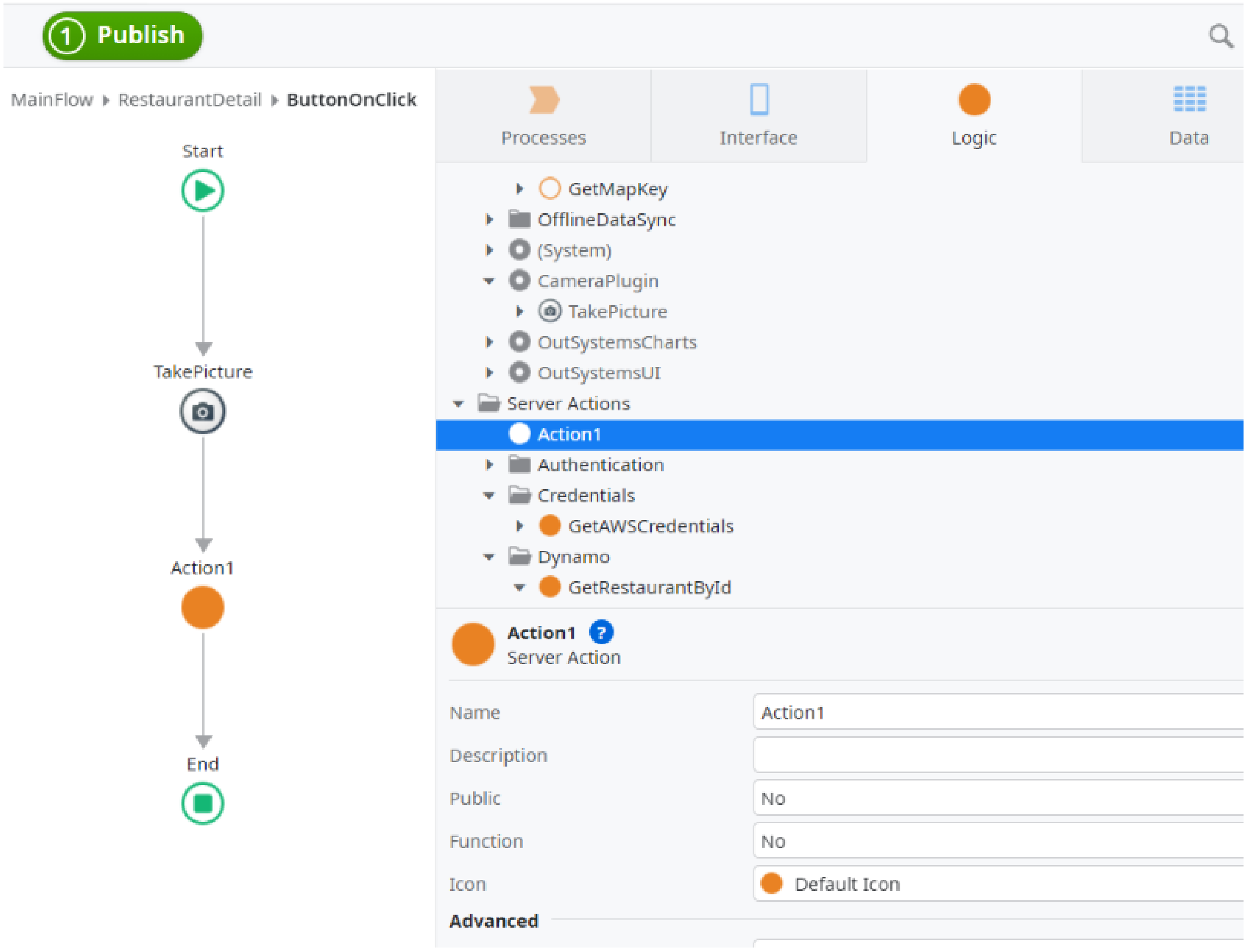
On Click: ButtonOnClick

30) On this new action click on the "**plus button**" in the middle of the flow and select a "Client Action". Then select the "**CameraPlugin**" dependency, and select the

"TakePicture" action.



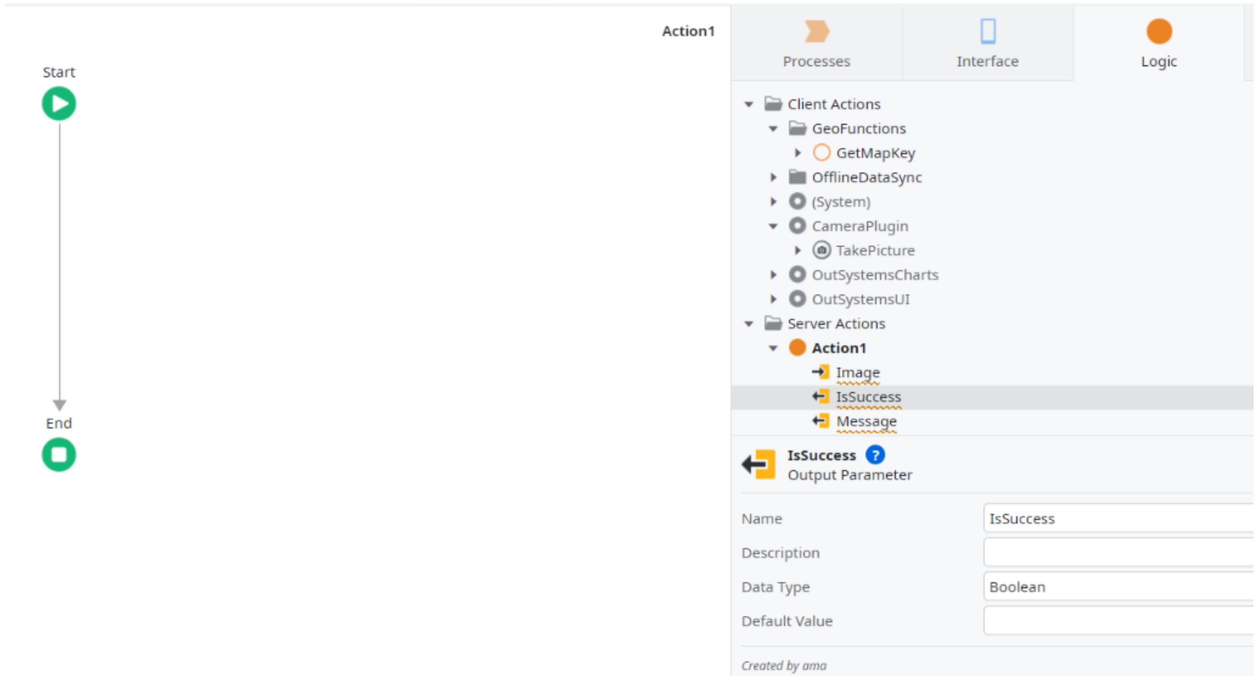
31) Now do the same but to create a new Server Action



Name	Action1
Description	
Public	No
Function	No
Icon	Default Icon

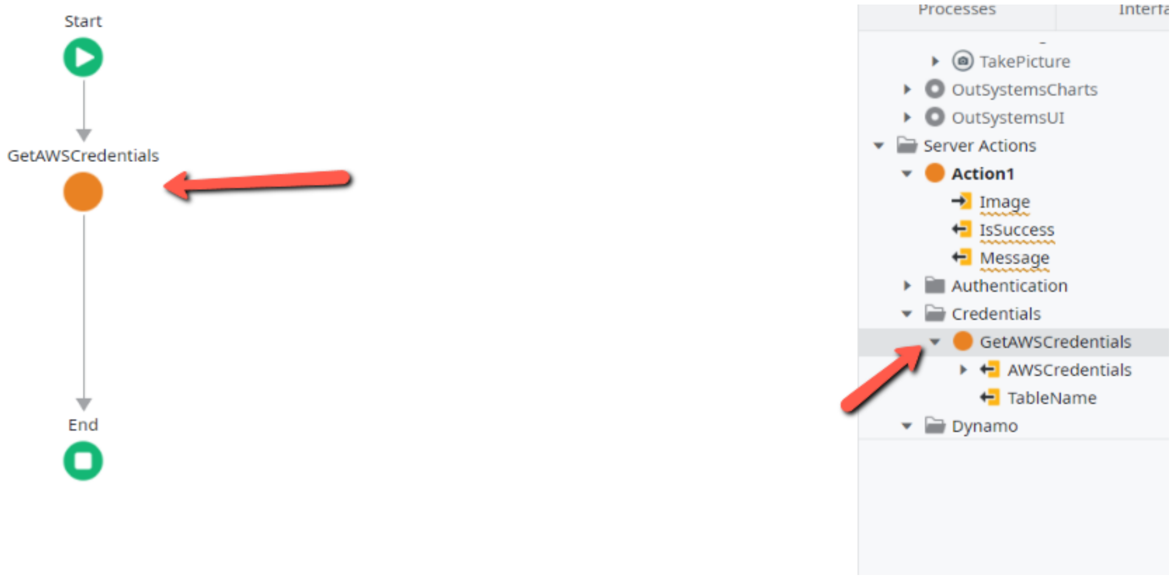
Rekognition + S3

32) Let's build the server action to check if the pictures are from restaurants or food, before uploading them to S3. Open the Action1 from the logic tab and add an input parameter called Image with the type Binary. Add an output parameter called IsSuccess with the type boolean, and another output Text variable called Message



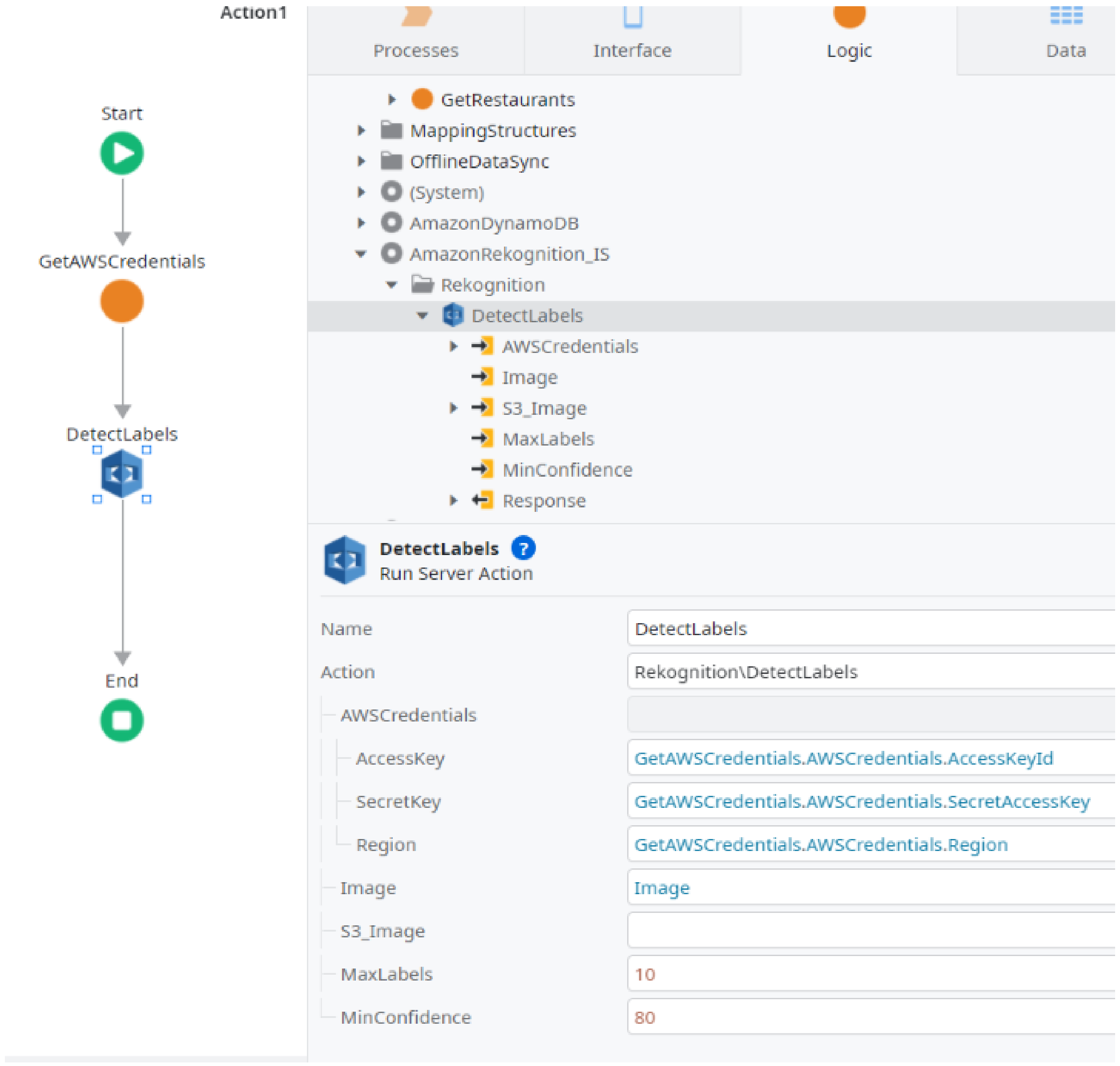
Name	IsSuccess
Description	
Data Type	Boolean
Default Value	

33) In the Logic tab, from the Credentials folder, drag the GetAWSCredentials action. This action returns the keys needed to use the AWS IAM user with access to the necessary APIs



34) From the AmazonRekognition_IS, drag the DetectLabels action. If you hover with the mouse on the left of the AWSCredentials attribute you'll be able to click on a plus sign.

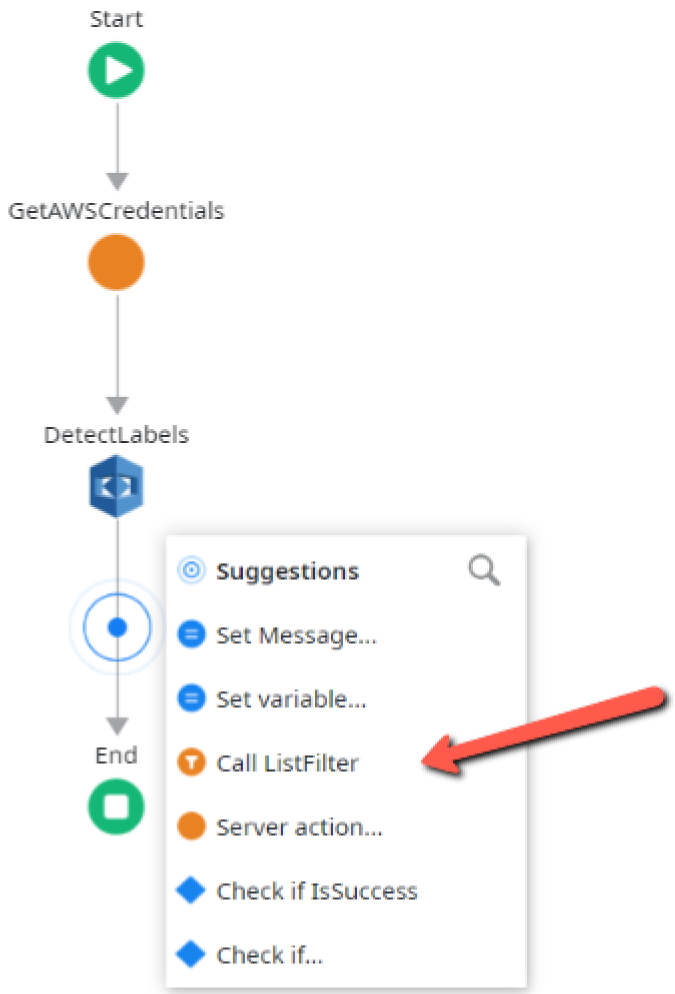
Map the outputs from the **GetAWSCredentials** action to the **AccessKey**, the **Secret Key** and the **Region**. Also pass to the Image attribute the **input image**.The **max labels** define 10 and the **MinConfidence** define 80. This way the **DecLabels** will return a list of a maximum of 10 labels all with confidence superior to 80%



DetectLabels
Run Server Action

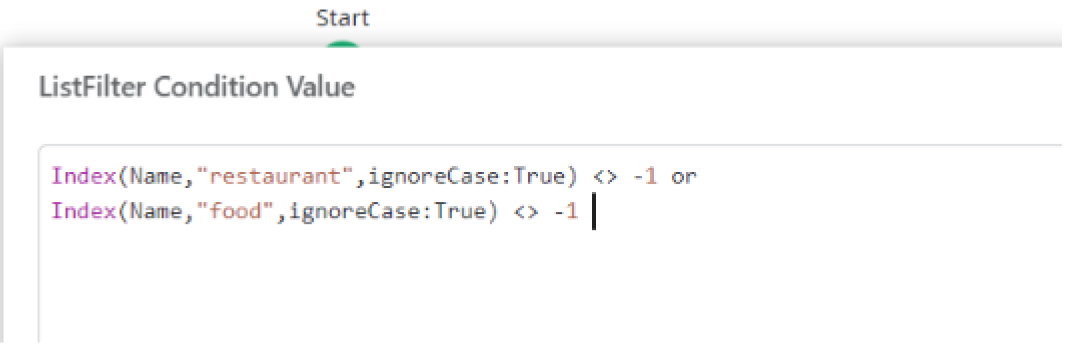
Name	DetectLabels
Action	Rekognition\DetectLabels
AWSCredentials	
AccessKey	GetAWSCredentials.AWSCredentials.AccessKeyId
SecretKey	GetAWSCredentials.AWSCredentials.SecretAccessKey
Region	GetAWSCredentials.AWSCredentials.Region
Image	Image
S3_Image	
MaxLabels	10
MinConfidence	80

35) Click on the blue dot after that action to get some AI suggestions and put a list filter. It will have pre-selected the SourceList.



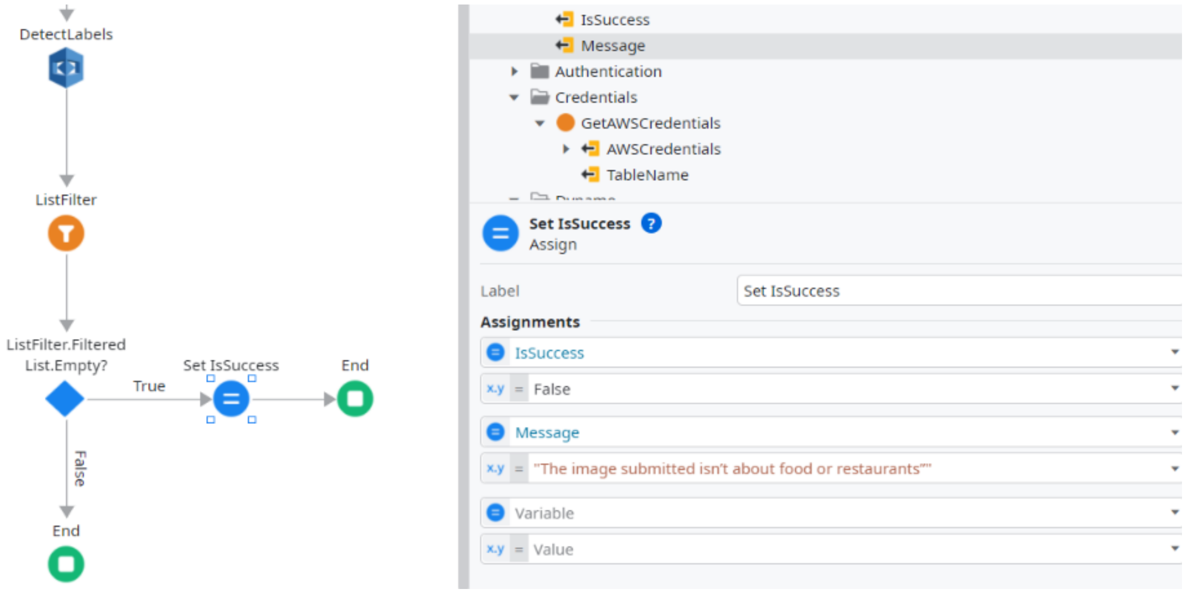
36) For the condition we are going to check if any of the Labels are for “**Restaurant**” or for “**food**” to make sure that the uploaded image is indeed useful to add to the restaurant. To compare the text we are going to use the Index function that returns “-1” if the text passed is not found: On the condition attribute put “**Index(Name,"restaurant",ignoreCase:True) <> -1 or**

Index(Name,"food",ignoreCase:True) <> -1”



```
Index(Name,"restaurant",ignoreCase:True) <> -1 or
Index(Name,"food",ignoreCase:True) <> -1
```

37) To check if Rekognition was able to identify a label of food or restaurant with enough confidence we just need to check if the list filtered is not empty. Put an if statement, where you check for “**ListFilter.FilteredList.Empty**” and pull a True branch from the if, that connects to an Assign that defines the IsSuccess as false and defines the message as “The image submitted isn’t about food or restaurants”



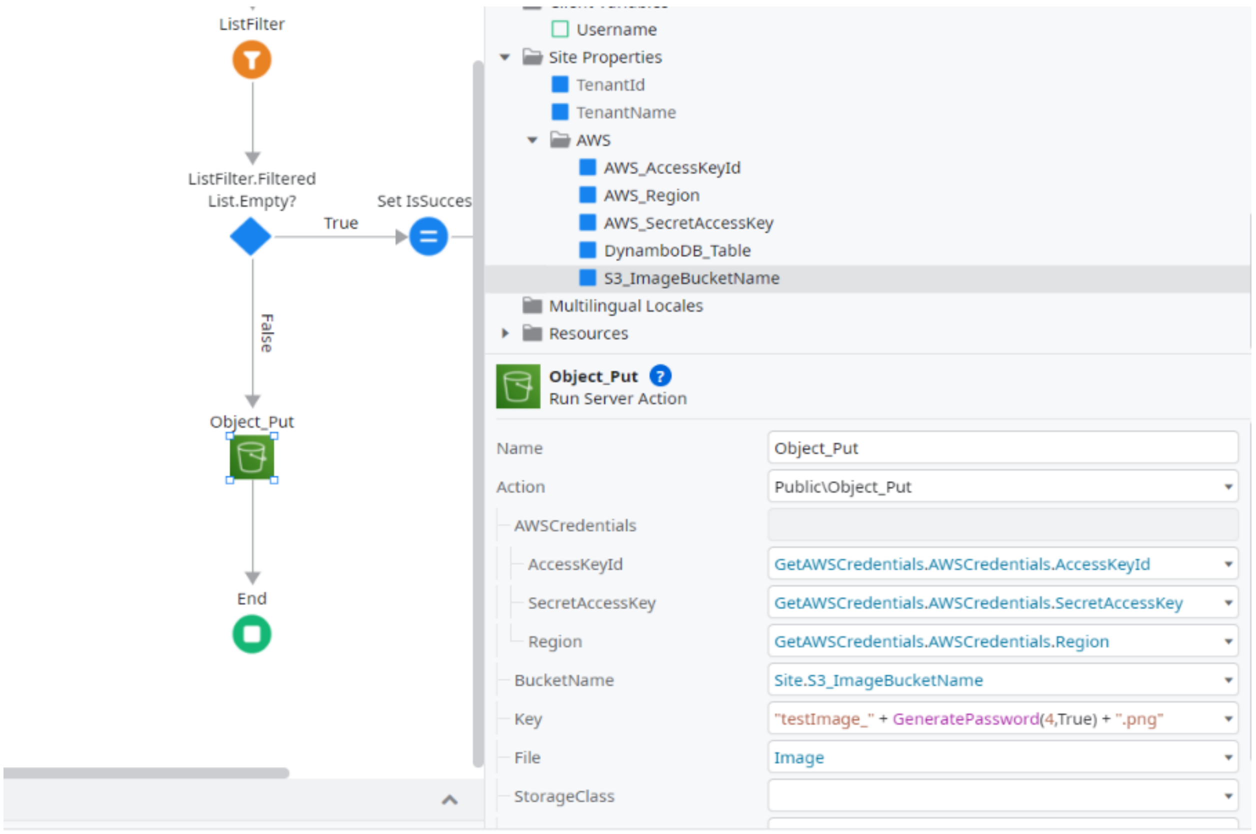
Label: Set IsSuccess

IsSuccess = False

Message = "The image submitted isn't about food or restaurants""

38) On the false branch, you can now drag the “**Object_Put**” action from the **AmazonS3_IS** dependency. Map the **AWSCredentials** as you already have done for the “**DetectLabel**”. Get the bucket name by putting the site property **Site.S3_ImageBucketName**. For the key generate a name for the imagefile in the bucket and you can use the “**GeneratePassword()**” function to generate a random text for the images to have different identifiers. You can give the key **“testImage_” + GeneratePassword(4,True) + “.png”** to get a file name like “testImage_2f43.png”. Fine assign the input image to the

"**file**" attribute

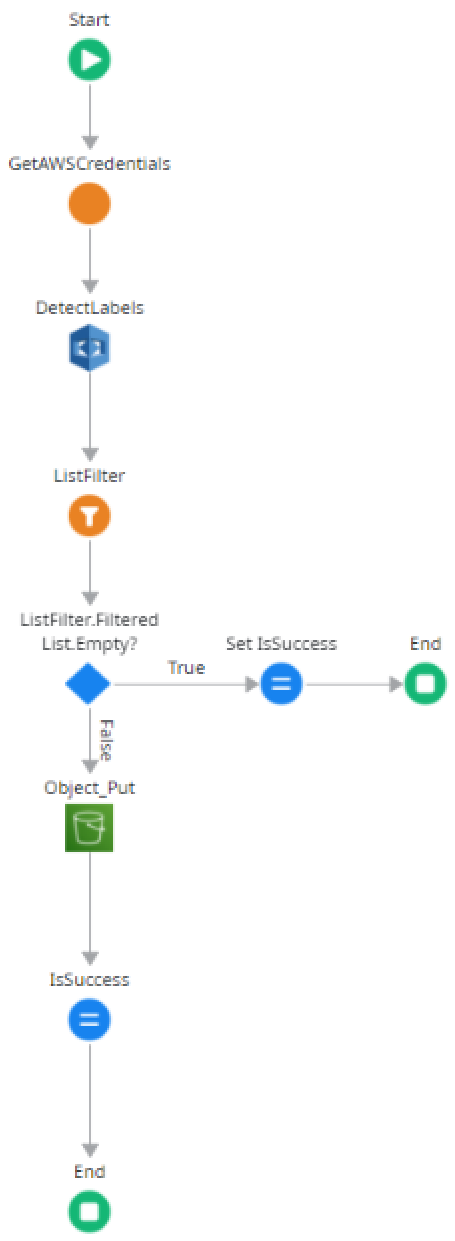


Username
Site Properties
TenantId
TenantName
AWS
AWS_AccessKeyId
AWS_Region
AWS_SecretAccessKey
DynamboDB_Table
S3_ImageBucketName
Multilingual Locales
Resources

Object_Put ?
Run Server Action

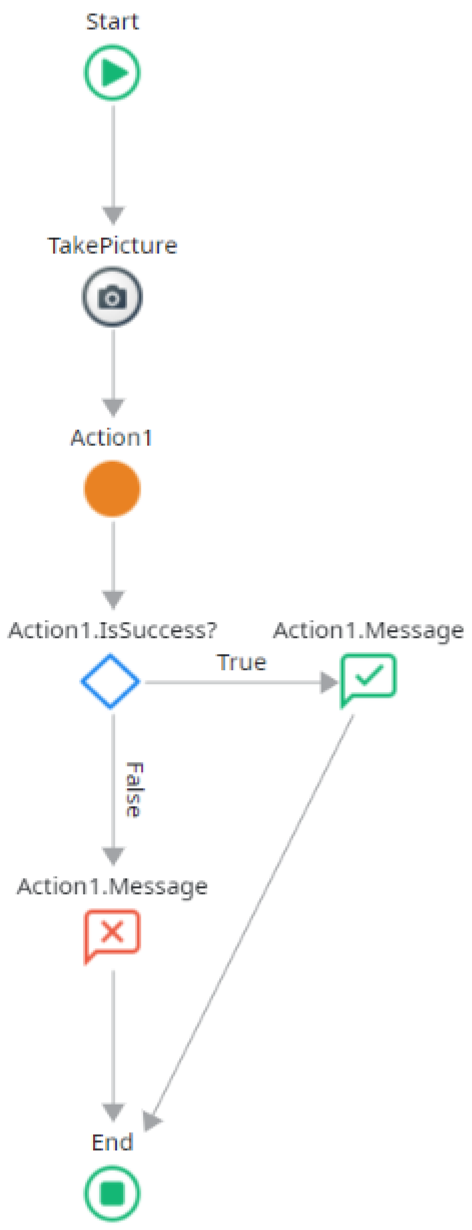
Name	Object_Put
Action	Public\Object_Put
AWSCredentials	
AccessKeyId	GetAWSCredentials.AWSCredentials.AccessKeyId
SecretAccessKey	GetAWSCredentials.AWSCredentials.SecretAccessKey
Region	GetAWSCredentials.AWSCredentials.Region
BucketName	Site.S3_ImageBucketName
Key	"testImage_" + GeneratePassword(4,True) + ".png"
File	Image
StorageClass	

39) To finish the action, get an “**Assign**”, put the **IsSuccess** as true and the server action is finished. Assign the Message “Upload successful” to the **Message** output



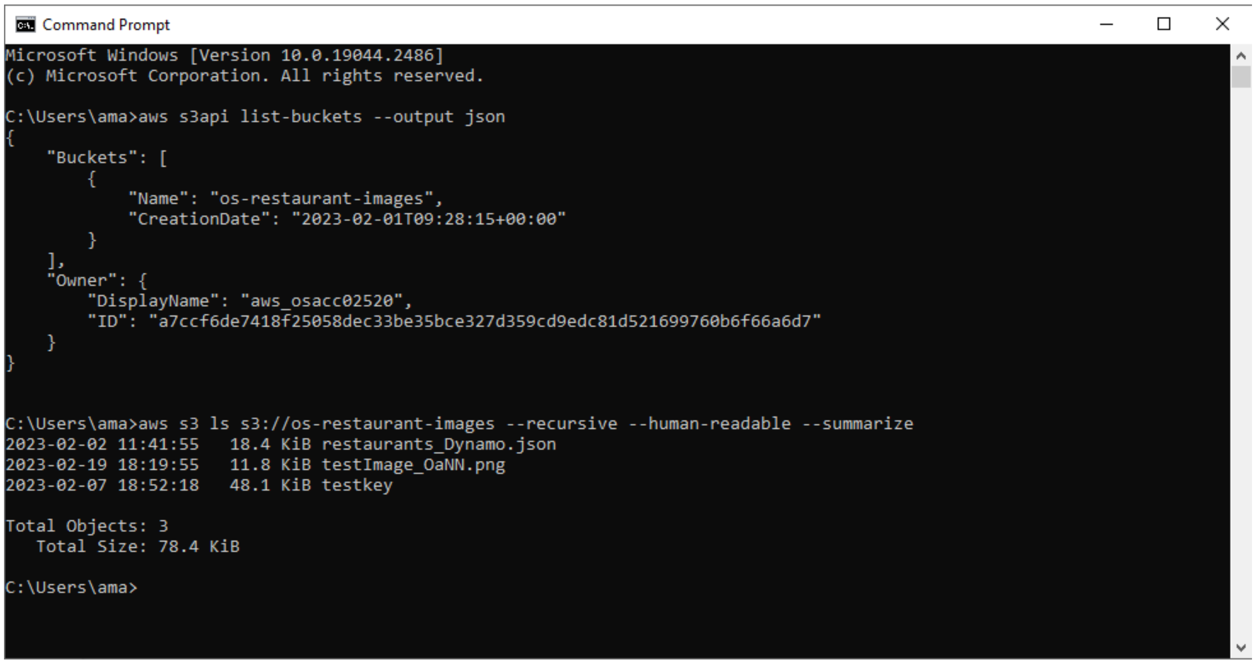
Finishing up

40) Now go to the "**ButtonOnClick**" client action on the **RestaurantDetail** screen and in the image attribute of this created action, put the image captured of the "**TakePicture**" action. No add an **IF** checking the **IsSuccess** boolean from **Action1**. In the true branch give a positive Message with the **message** output of the server action. In the false branch do the same but with a red message.



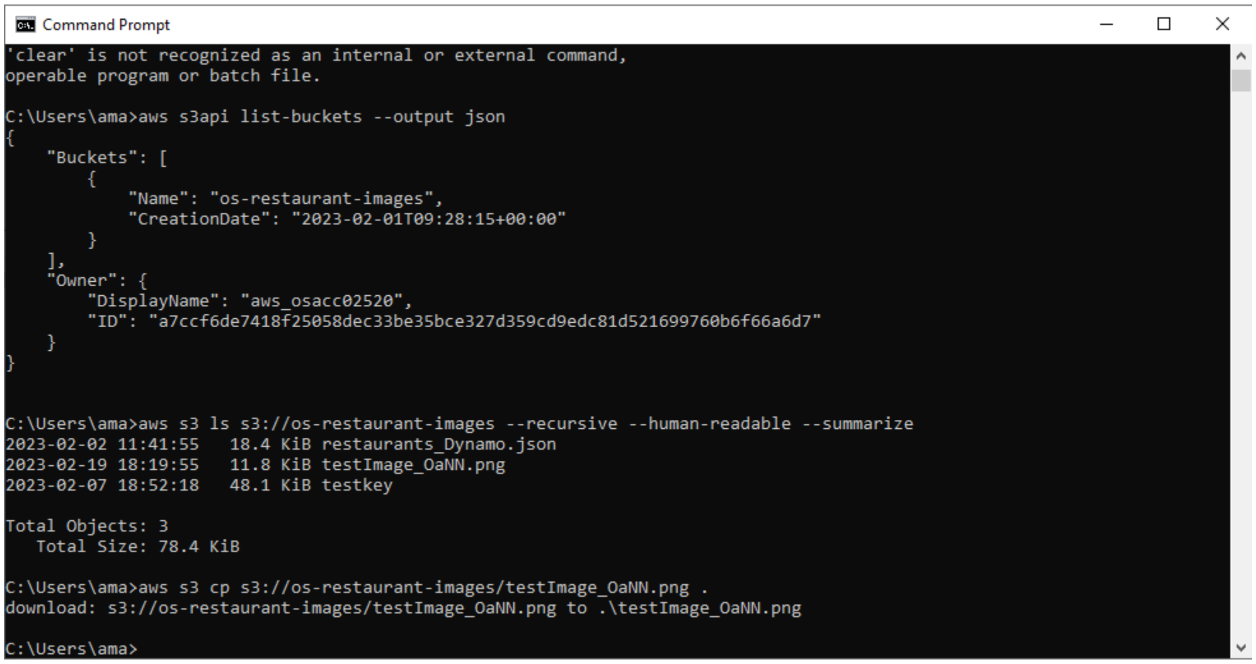
41) To test it out use the application, try to take a picture of a blend background and see it we get the negative message. Then try to take a picture of food or a restaurant setting and it should give the positive message of a successful upload.

42) Get back to the AWS CLI. You can run a command to check the content of the bucket
aws s3 ls s3://os-restaurant-images --recursive --human-readable --summarize
Run before and after you take a successful picture.



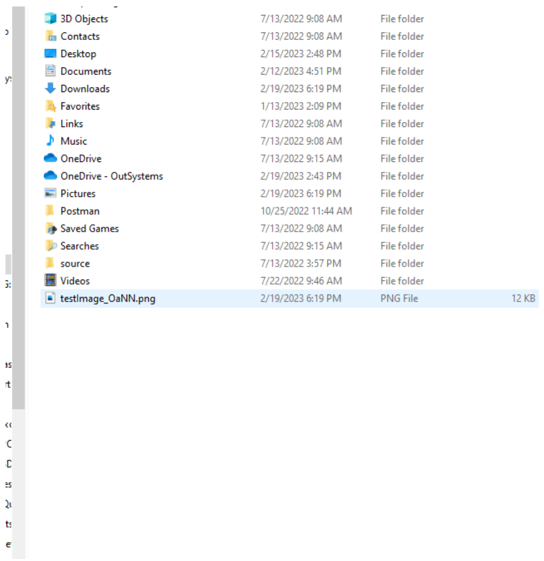
```
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ama>aws s3api list-buckets --output json
{
    "Buckets": [
        {
            "Name": "os-restaurant-images",
            "CreationDate": "2023-02-01T09:28:15+00:00"
        }
    ],
    "Owner": {
        "DisplayName": "aws_osacc02520",
        "ID": "a7ccf6de7418f25058dec33be35bce327d359cd9edc81d521699760b6f66a6d7"
    }
}


C:\Users\ama>aws s3 ls s3://os-restaurant-images --recursive --human-readable --summarize
2023-02-02 11:41:55   18.4 KiB restaurants_Dynamo.json
2023-02-19 18:19:55   11.8 KiB testImage_OaNN.png
2023-02-07 18:52:18   48.1 KiB testkey

Total Objects: 3
   Total Size: 78.4 KiB

C:\Users\ama>
```

43) You can copy the file to your desktop using the following command as an example:
aws s3 cp s3://os-restaurant-images/testImage_OaNN.png .

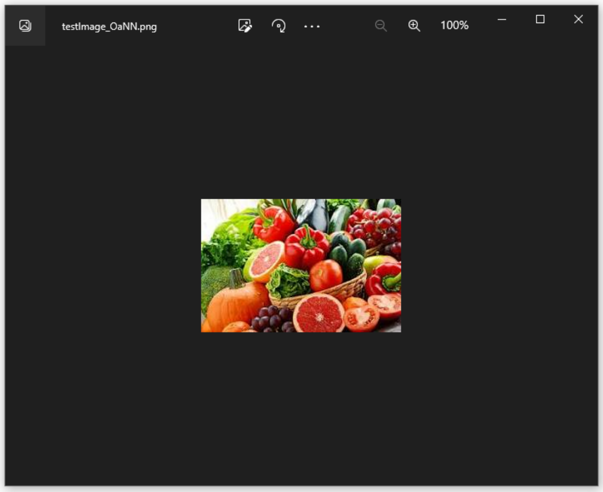


```
'clear' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\ama>aws s3api list-buckets --output json
{
    "Buckets": [
        {
            "Name": "os-restaurant-images",
            "CreationDate": "2023-02-01T09:28:15+00:00"
        }
    ],
    "Owner": {
        "DisplayName": "aws_osacc02520",
        "ID": "a7ccf6de7418f25058dec33be35bce327d359cd9edc81d521699760b6f66a6d7"
    }
}


C:\Users\ama>aws s3 ls s3://os-restaurant-images --recursive --human-readable --summarize
2023-02-02 11:41:55   18.4 KiB restaurants_Dynamo.json
2023-02-19 18:19:55   11.8 KiB testImage_OaNN.png
2023-02-07 18:52:18   48.1 KiB testkey

Total Objects: 3
   Total Size: 78.4 KiB

C:\Users\ama>aws s3 cp s3://os-restaurant-images/testImage_OaNN.png .
download: s3://os-restaurant-images/testImage_OaNN.png to .\testImage_OaNN.png

C:\Users\ama>
```

44) Then you can open your file browser in the desktop and show the copied image



3D Objects	7/13/2022 9:08 AM	File folder	
Contacts	7/13/2022 9:08 AM	File folder	
Desktop	2/15/2023 2:48 PM	File folder	
Documents	2/12/2023 4:51 PM	File folder	
Downloads	2/19/2023 6:19 PM	File folder	
Favorites	1/13/2023 2:09 PM	File folder	
Links	7/13/2022 9:08 AM	File folder	
Music	7/13/2022 9:08 AM	File folder	
OneDrive	7/13/2022 9:15 AM	File folder	
OneDrive - OutSystems	2/19/2023 2:43 PM	File folder	
Pictures	2/19/2023 6:19 PM	File folder	
Postman	10/25/2022 11:44 AM	File folder	
Saved Games	7/13/2022 9:08 AM	File folder	
Searches	7/13/2022 9:15 AM	File folder	
source	7/13/2022 3:57 PM	File folder	
Videos	7/22/2022 9:46 AM	File folder	
testImage_OaNN.png	2/19/2023 6:19 PM	PNG File	12 KB



45) You are finished with the Demo. But from the upload of the image to S3 you can extend your solution in AWS. You could trigger a Lambda function from the S3 that could process the upload image, creating a thumbnail image and normalizing it and that could change it to a repository where all user uploaded pictures are. Then the lambda could add it to the DynamoDB database