

PowerApps: Creating small business applications with less effort & time

When working on different projects at various client sites, there were different occasions where the client business identifies a gap in its processes and highlights the need for a small application to be able to solve this.

For example, we encounter many situations where companies are still using Excel as a data storage or reporting tool, Macros or Access Forms. Or business problems that cannot be addressed by off-the-shelf software. Likewise, there are many other instances, even when working towards a Modern Data Warehouse, where data (master-data) need to be inputted in the database and controlled or maintained.

Up till recently, when this happened, no matter the size of the application required, the client would probably ask developers for a brief analysis and idea of what the project would involve if that application was to be developed. And the developers would probably explain that for a small mobile application, the project could be approached by focusing on the Backend, Frontend and Testing as the main features. For the backend we will have different user stories probably covering the server-side logic, data integrations, data storage, user management and version control.

Then for frontend, we need to factor for User-Experience/ User-Interface design, then development and also things like caching.

When we are in a position to produce a deliverable or a minimum viable product, we would need to start testing throughout the different project sprint. And that Testing can be probably split in Functional testing, Cross-platform testing, Unit Testing, User Acceptance Testing, Performance Testing, Security Testing, UX/UI testing, compatibility testing, and the list goes on.

Then we need to identify how we are going to take care of the support and maintenance of the application, especially if there are no in-house developers, and this could include risks like for example future compatibility with operating systems and browsers.

And then the client would be like, “Great dear developer – it seems like you are in control of this, you are making it sound simple and maybe we can consider you with building this application. But before that, obviously we need to see the cost and have a rough idea of the time and effort involved. Any ideas on this? Would it be ready by next week?” And probably this was this point for many years, where we would diplomatically need to think how to tell the client to sit down, have a bit of coffee, maybe some small talking. And then try to explain that well, there is work that needs to be done, and this could take weeks to deliver and many iterations or sprints – depending on the scope and the team size could easily be a minimum of 10 weeks and a several hundreds or thousands as a price tag.

And if by that time, the client is still sitting still on the chair, you will probably be faced with different face expressions and things like “Oh, is it that long and costly?”, “It is just a small application”, “I would need to get more budget approval then I thought for this, it will take us time to get sign-off, if we manage to get there!”, “Not sure if it will remain viable by the time we get sign-off!”

And this happened to me on many occasions, and I don’t think it’s just for me.

But technically, even for a small mobile application – should we have to investment this much time, effort and cost to deliver it? And even if we put the time, effort and cost factor aside, should someone be an experienced developer in order to be able to produce a solution?

Would it be helpful to have a way to facilitate all this process and be able to produce a solution with less time and effort that might be enough to meet the business requirements through a small application?

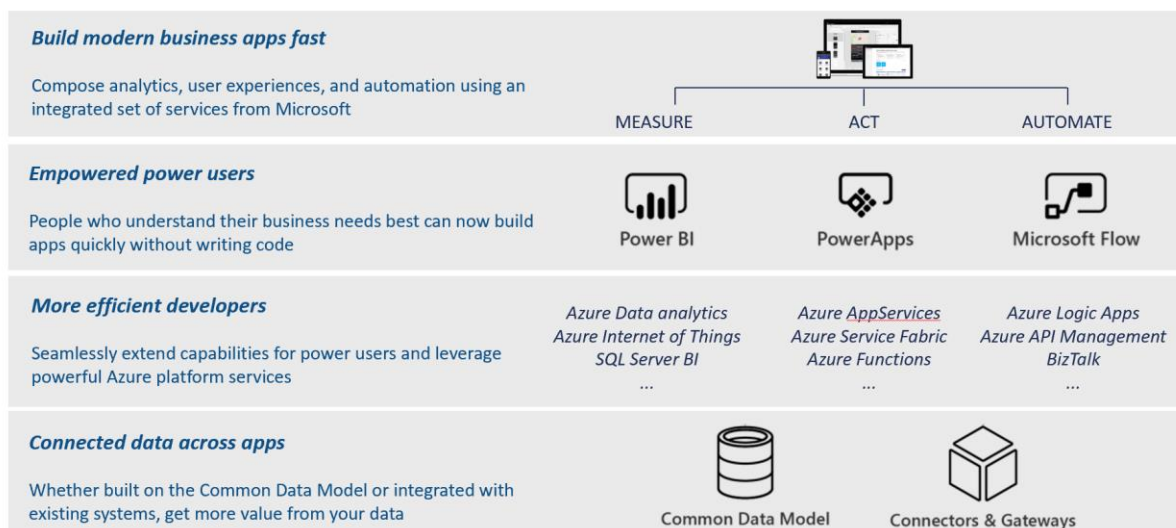
And the scope of this session, is to explain how we can drastically improve this process.

After we again we faced one of these situations, where a client required a small business application, we opted to use a new technology that was lately released by Microsoft – and that was PowerApps.

And the question that probably comes up to the mind of many is, but what is PowerApps?

PowerApps is a LOW PROGRAMMING and RAPID DEVELOPMENT tool, that allows NON-TECHNICAL & TECHNICAL PERSONNEL to BUILD THEIR OWN APPS, intended mostly for MOBILE & TABLET using a combination of VISUALS and pre-defined FUNCTIONS, TEMPLATES or CONNECTIONS.

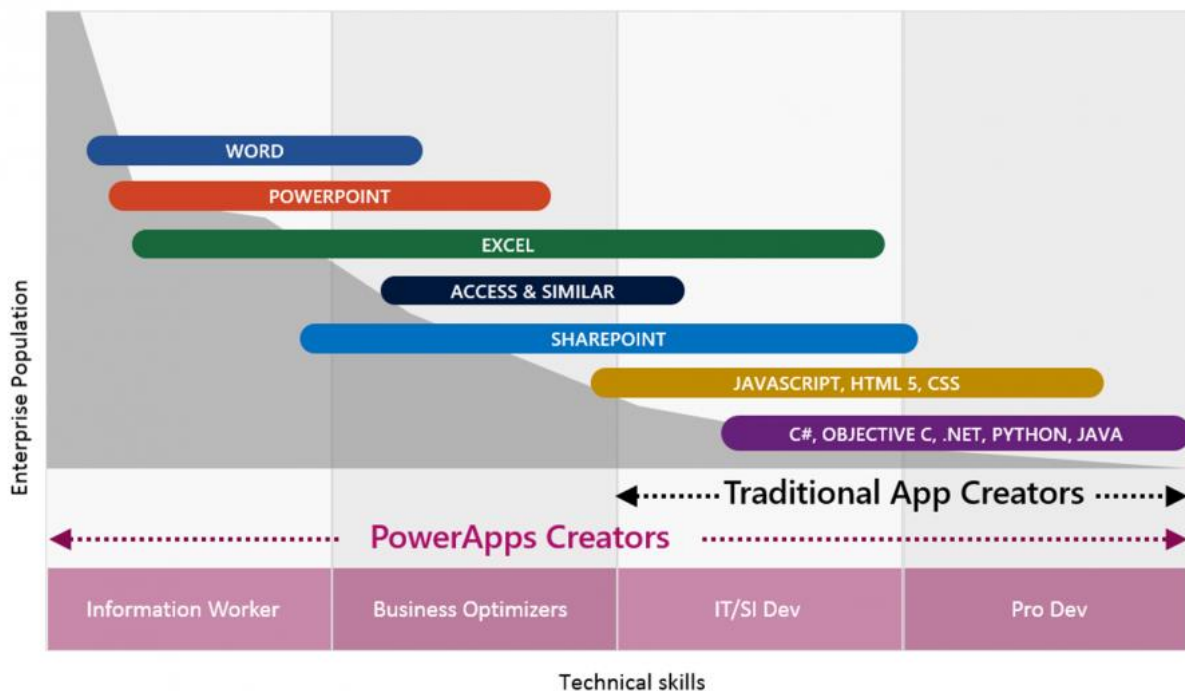
PowerApps is also part of the Power Platform, which incorporates three Microsoft technologies and is described here:



Picture credit: <https://community.dynamics.com/partner/b/msuspartner/posts/office-365-partner-community-microsoft-powerapps-and-microsoft-flow>

And if you were to ask me about at least 3 ways through which PowerApps would be beneficial in such a situation, I could easily pinpoint these out to you.

Firstly...



Picture credit: <https://blogs.msdn.microsoft.com/mvpawardprogram/2017/04/04/part-1-powerapps-and-flow/>

- 1) With the drag and drop facility, and some familiarization on online material for the logic, YOU can build the application yourself, without having to have experience in other programming languages like ASP.NET MVC, CSS, Java Script, Gzamarin (Xamarin) or the likes. It does not mean that if you know these you cannot still use PowerApps, or that it would not in any way help – BUT it is not necessary for you to produce a small application.

Second advantage...

- 2) You do not need to think also for infrastructure, hosting, excessive manual adaptations for different platforms and such things. PowerApps platform, in itself will produce a solution that can be reachable by the shared with users or now even by public users, when the publish button is clicked. A URL is easily generated and that can be accessible from your own browser. Your focus will be on the design and development of the application and not much more.

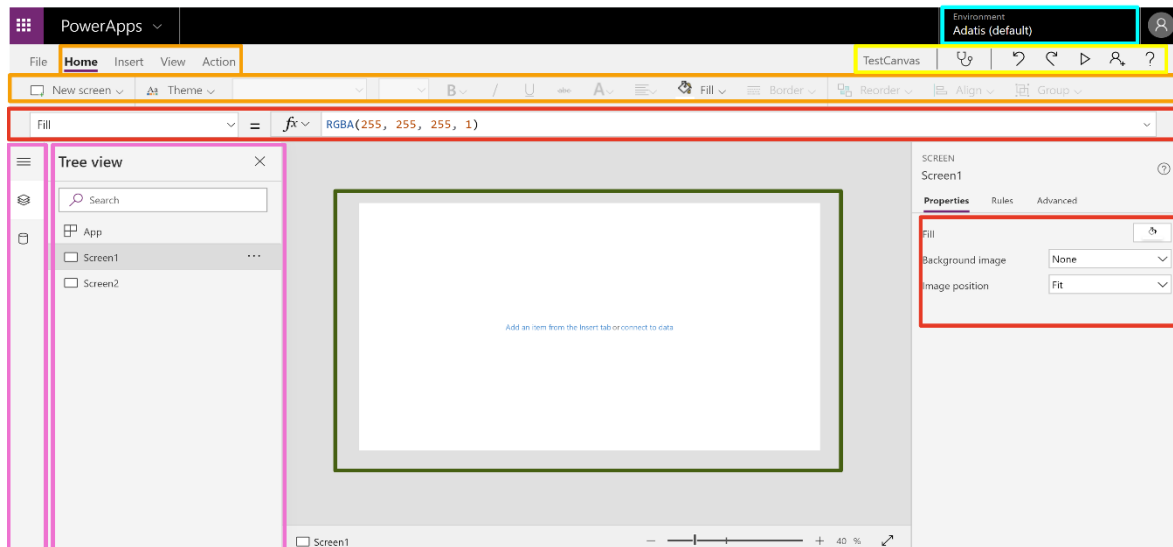
A third advantage...

- 3) PowerApps integrates with many connectors, out-of-the-box, just with some clicks of buttons or sign-ins.

API connections with various platforms, namely:

- Social Media
- Customer Relationship Management Software
- Enterprise Resource Planning
- Azure services
- Marketing Automation Platforms
- Custom APIs

These are only 3 points – but probably as you start working with PowerApps, you will find others as well.



Here is what development in PowerApps look like when building a canvas application.

- 1) Orange: Depending on what we choose from the upper functions highlighted in orange (Home/ Insert/ View/ Action), we will have a list of different allowed features. For instance, in the Home tab, we will be able to create new screens, formatting, adjusting layer ordering etc.
- 2) Green: In the green area, we can see the canvas, which we will be using to design our application on. What we see there, is what we are going to get on that screen in that application (obviously as long as we do not set hidden objects etc.) So, any buttons, images, tables, text input controls are going to be placed here.
- 3) Red: Every application, its screens, and object within a screen would have a number of attributes that one can set or program. We can see that there is either a more visual layout for this on the right or could set the same attributes from a ribbon above the canvas. It is a matter of preference and might also depend on what attribute you are setting.
- 4) Light Blue: That indicates the environment name where the application is hosted and saved on. Think of this as a workspace. You might have multiple applications saved on different environments, and different people might have access to the environment itself (or different rights on the environment). For example, in a development scenario, you might have 3 environments setup for one project, one for Development, one for UAT and one for Production. You can export and import applications between environments.
- 5) Yellow: Here is the application name, redo/undo and other functions like an Error Checker and a Play function for the application to see the application from a non-development perspective (and in action).
- 6) Pink: Depending on what you choose in the left pink box, you will have related options in the other pink box. Here we are choosing a tree view which allows us to shift between different

screens of the applications. But we can also choose to see a list of connections that we have from the cylinder in the first pink box.

Exercise:

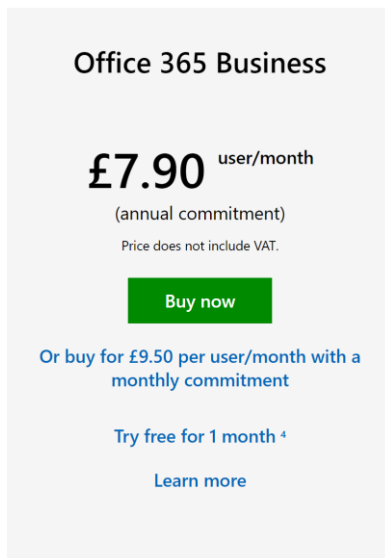
Here is an exercise to create a small PowerApps application from scratch, so that you can see the entire process from start to finish.

Creating an Office 365 trial account

(Skip this section if you already have access to an Office 365 account)

If you do not have an Office 365 account already, you can create a trial account through this link:

<https://products.office.com/en-gb/compare-all-microsoft-office-products?&activetab=tab:primaryr2>



The screenshot shows the Office 365 Business pricing card. At the top, it says "Office 365 Business". Below that, the price is listed as "£7.90 user/month" with "(annual commitment)" underneath. A small note states "Price does not include VAT." There is a green "Buy now" button. Below the button, it says "Or buy for £9.50 per user/month with a monthly commitment" in blue text. Further down, it says "Try free for 1 month ⁴" in blue text, followed by a "Learn more" link in blue text.

Under Office 365 Business, click “Try free for 1 month”.

3 Create your business identity



To set up your account, you'll need a domain name. [What is a domain?](#)

You'll probably want a custom domain name for your business at some point. For now, choose a name for your domain using **onmicrosoft.com**

yourbusiness .onmicrosoft.com

onmicrosoft.com is available.

Check availability

Next

4 Get Office

Save this info. You'll need it later.

Sign-in page

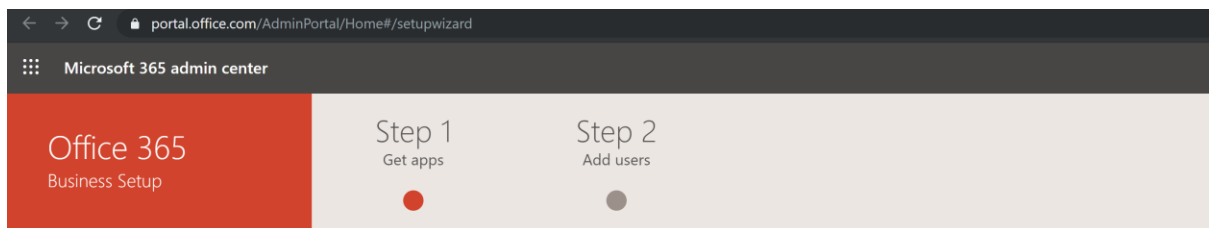
<https://www.office.com/>

Your user ID

@ .onmicrosoft.com

Go to Setup

Remember your user ID as you will need it to log in PowerApps later.



Install your Office apps

Install Office apps for yourself so you can take full advantage of your subscription. Selecting Install now will assign a subscription to you, if you don't have one already.



Microsoft Office Professional Plus

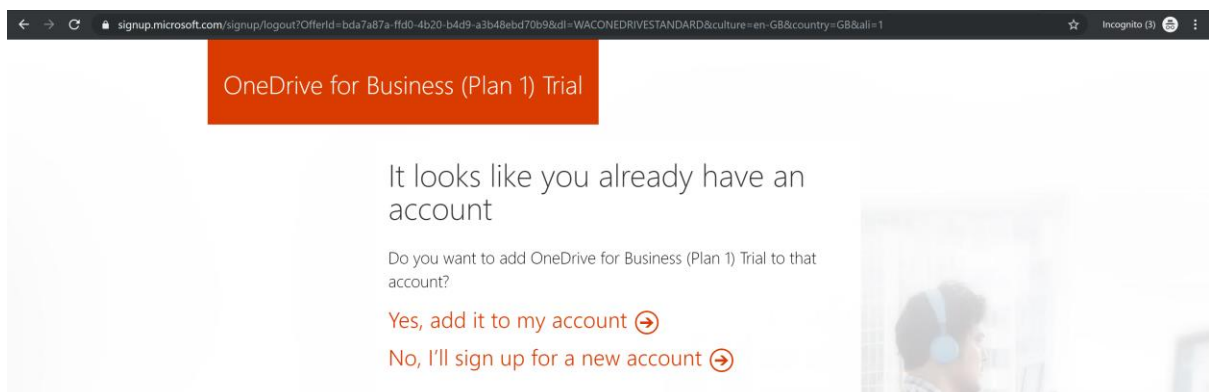
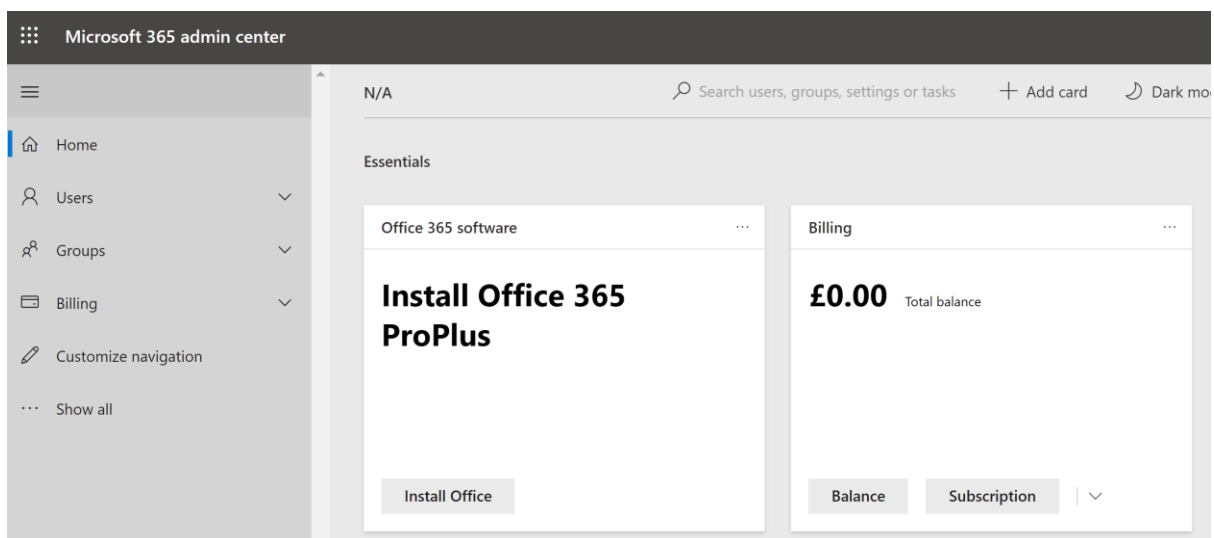
Install the following apps on your computer: Word, Excel, PowerPoint and Outlook.



[Install now](#) ▾

Want to install the apps later? Assign a license to yourself and go to the Office software card on the Admin center home page to find your download links.

Next ⌚ [Exit and continue later](#)





Hi [redacted]

We need some information to set up your free trial.

Choose your country to begin

United States

Microsoft may send occasional emails. You can unsubscribe any time.

By choosing **Start my trial**, you agree to the PowerApps terms of use and Microsoft privacy statement.

Cancel Start my trial

We will be continuing on this further after determining the connection, in the PowerApps section further down.

To Save to SQL

If you have an existing Azure SQL instance running, it might be worth creating a blank database and running this script within it for this exercise.

```
CREATE SCHEMA [People]
GO

CREATE TABLE [People].[Employees](
    [RecordID] [int] IDENTITY(1,1) NOT NULL,
    [FirstName] [varchar](max) NULL,
    [LastName] [varchar](max) NULL,
    [Email] [nvarchar](max) NULL,
    CONSTRAINT [PK_PeopleEmployees] PRIMARY KEY CLUSTERED
    (
        [RecordID] ASC
    )WITH (STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Please bear in mind that on-premise SQL databases would require PowerApps premium license.

To Save to Excel

As a start, I would suggest to avoid building an application based on an Excel. However if you want to run a test on how PowerApps works and you do not have access to a SQL database, and you do not have a PowerApps Premium license (which would have allowed CDS), then it could serve as a means of completing this exercise (for the sole purpose of understanding PowerApps).

Go to <https://onedrive.live.com/> and login with the business user address you just created.

Personally, I tend to use OneDrive App to be able to access files from Windows. To do this you might need to download the application: <https://onedrive.live.com/about/en-gb/download/>

Alternatively, if you have it downloaded already, go to Start OneDrive.

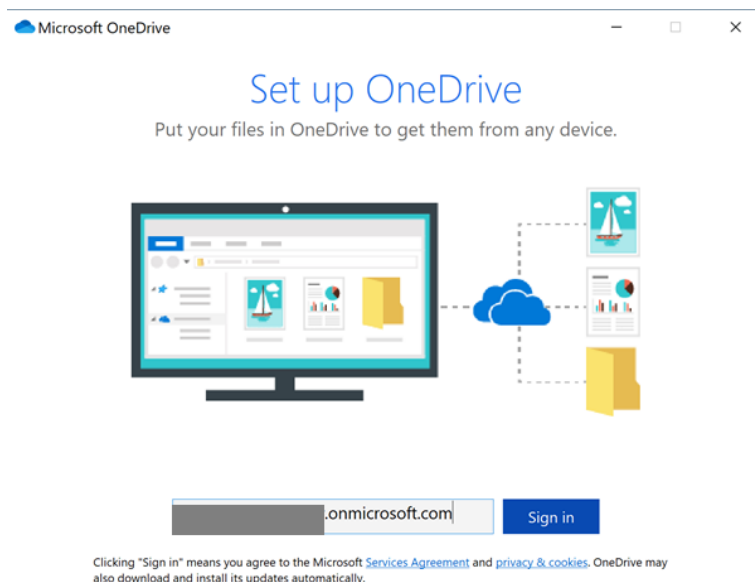
OneDrive for Windows

You're running Windows 10, so OneDrive is already installed on your computer. Don't forget to get the app and take your files on the go!

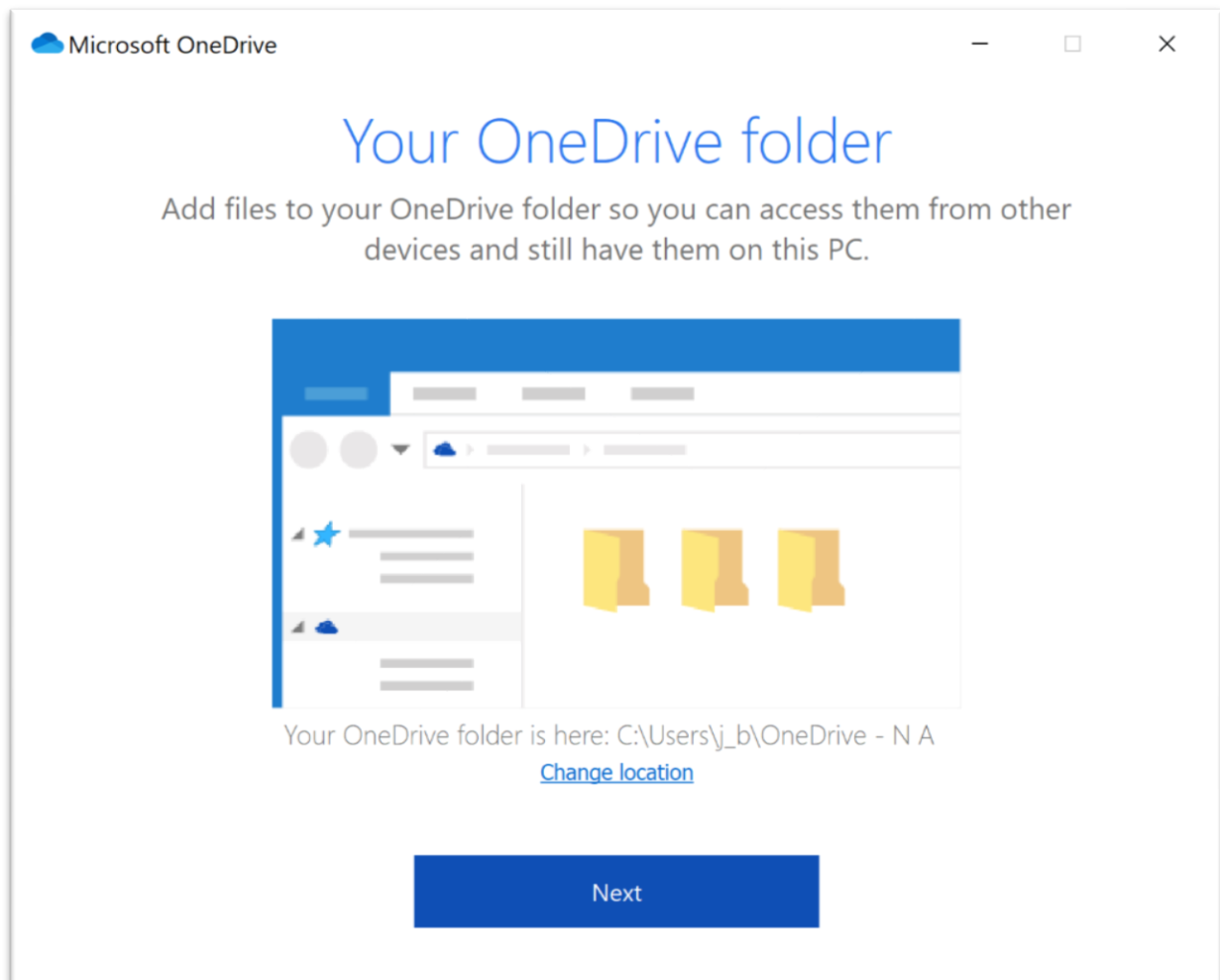
Start OneDrive

Need to reinstall? [Click here to download.](#)

Provide your business user ID, and sign in:



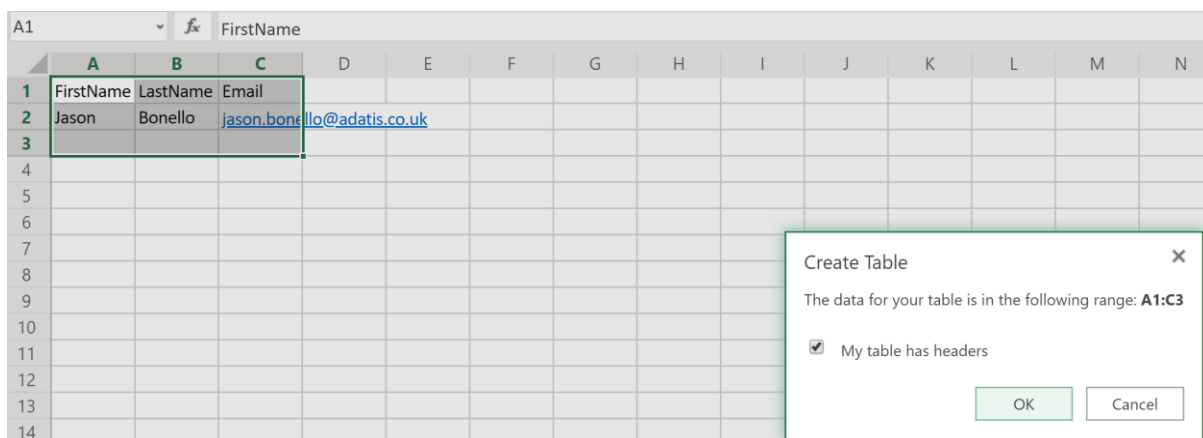
You eventually end up with a OneDrive folder within a physical location on your PC:



Create a new Excel sheet.

As headers, we will be using “FirstName”, “LastName” and “Email”, and I have filled in the second row with a value for each.

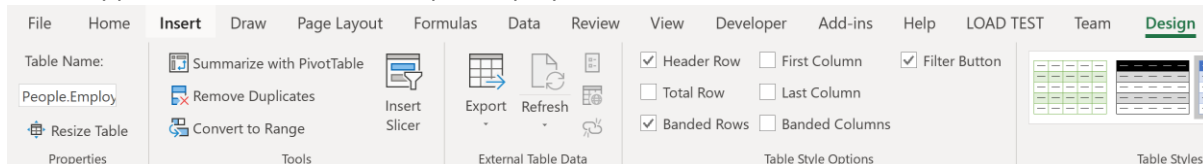
The important step here, is to change this to a Table (Insert > Table) – and tick the “My table has headers” checkbox.



This should make the excel looking similar to this:

	A	B	C
1	FirstName	LastName	Email
2	Jason	Bonello	jason.bonello@adatis.co.uk
3			

The Table Name in the Design ribbon will determine the table name that we are going to use in PowerApps, in this case I used “People.Employees”:

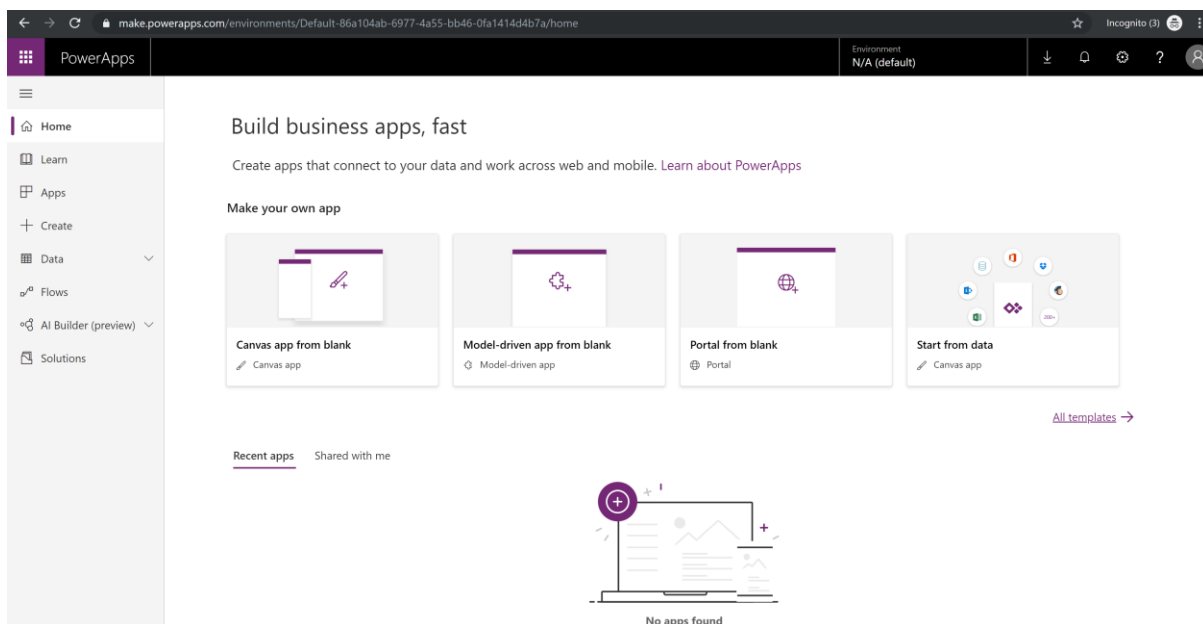


The Excel sheet is to be saved in the physical OneDrive folder, and in my case I named it as “Employees.xlsx”.

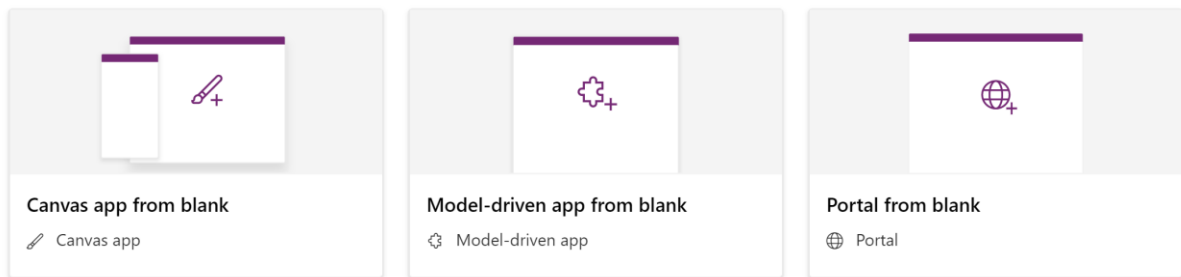
PowerApps – Create Application

Go to <https://powerapps.microsoft.com>

Login using the business user ID of the trial account just created.



Make your own app



In PowerApps, click on “Canvas app from blank”.

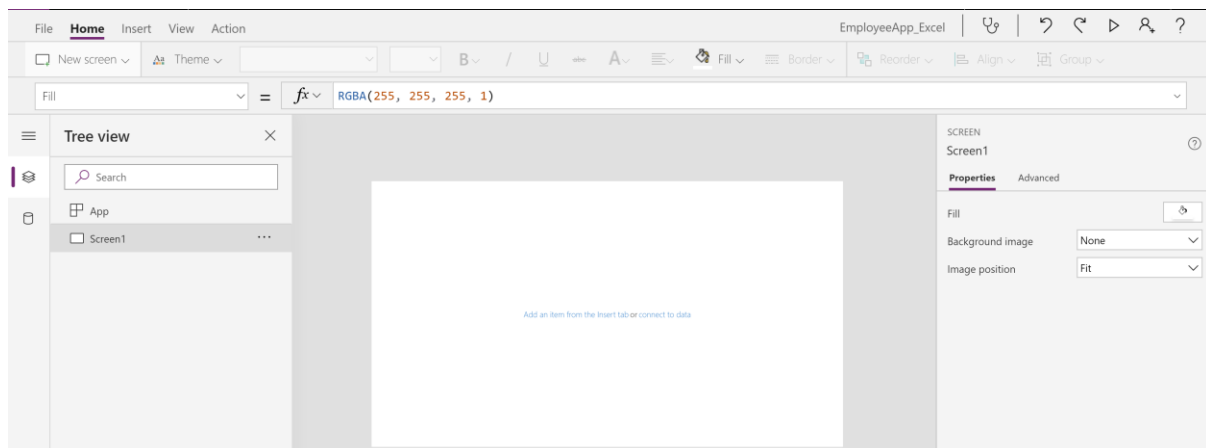
Give a name to the application, and for format you can adapt the screen to your preferred device. I will be using Tablet for this exercise. Then press “Create”.

Canvas app from blank

The screenshot displays the 'Canvas app from blank' configuration window. On the left, there is a large preview area showing a canvas with a pencil icon. Below this preview, a text box reads: 'Design the app you want, and connect it to hundreds of data sources.' At the bottom left of this section is a small icon and the text 'Canvas app'. On the right side of the window, there is a form with a close button (X) in the top right corner. The form includes a label 'App name *' above a text input field containing 'EmployeeApp'. Below this is a 'Format' section with two radio button options: 'Tablet' (which is selected) and 'Phone'. At the bottom right of the form are two buttons: a purple 'Create' button and a grey 'Cancel' button.

You might be faced with random help tips messages, for which you can click on Skip for now.

You should end up in the canvas.



Go to View, DataSources and under Connectors, click on “Show all connectors”.

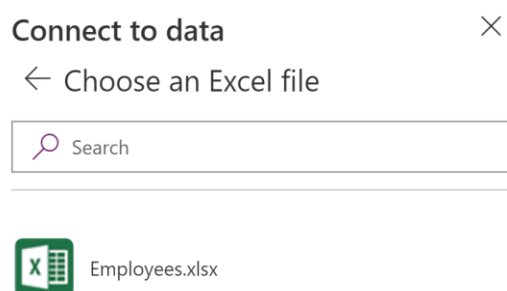
PowerApps - Adding DataSource

If you have chosen Excel as a datasource, then do the following in PowerApps:

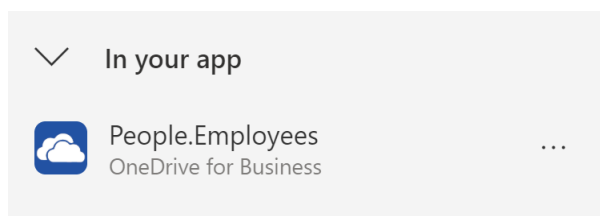
Choose OneDrive for Business.

Go to Create.

This will now show the Excel sheet for selection.



Choose this, and choose the table name.

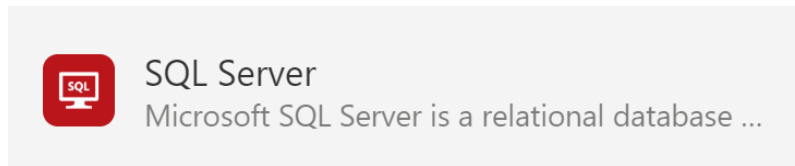


Once this has been added as a connection in PowerApps, you will notice that a PowerApps unique key has been added as an extra column in the existing Excel table.

	A	B	C	D	E
1	FirstName	LastName	Email	__Power	
2	Jason	Bonello	jason.bonello@adatis.co.uk	QDLV27jYkGA	

If SQL has been chosen as a datasource, the following should be done in PowerApps:

Choose SQL Server



Then choose “+ Add connection”

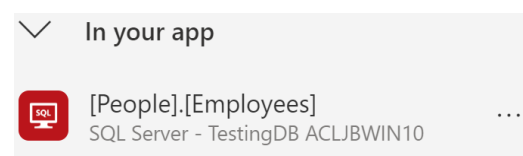
Fill in the connection details, and click on “Create”.

A screenshot of the 'Connect to data' dialog box for SQL Server. The dialog has a title bar with a close button (X). Below the title bar is a section with the SQL Server icon and text: 'SQL Server' and 'Microsoft SQL Server is a relational database management system developed by Microsoft. Connect to SQL Server to manage data. You can perform various actions such as create, update, get, and delete on rows in a table.' Below this are two radio buttons: 'Connect directly (cloud services)' (selected) and 'Connect using on-premises data gateway'. Under the selected option are four text input fields: 'SQL server name *', 'SQL database name *', 'Username *', and 'Password *'. At the bottom are 'Create' and 'Cancel' buttons.

Choose the earlier created table through the script and click Connect:

A screenshot of the 'Connect to data' dialog box, showing the 'Choose a table' screen. It has a title bar with a close button (X). Below the title bar is a back arrow and the text 'Choose a table'. There is a search bar with a magnifying glass icon and the word 'Search'. Below the search bar are two options: 'People.Employees' (selected with a purple checkmark) and 'Enter custom table name' (with an unchecked checkbox). At the bottom are 'Connect' and 'Cancel' buttons.

This will add the SQL table to the application:



Now as a note – which is outside the context of this presentation, but which is worth keeping in mind – bear in mind that access to the backend would be required before-hand (in order to be able to use the credentials that have the proper rights within the backend itself). On this note though, when working with confidential information, I strongly suggest that the right approach is taken when it comes to both backend access and PowerApps environment. For example, if you develop an application in the default, system-generated environment, add a SQL connection and share that application with someone else, even if only one table from that SQL has been used in that application, that user might end up having access to all other tables! And a way for the user to get access to other tables, is by creating a new app within their own default environment (which is common to the organization), and add that connection (which will be still retained with the hard-coded credentials) in their application. Then rather than choosing the same table they would be able to click on any other table and display the data from a Gallery. To avoid this, being on Plan 2 would allow you to create a separate environment (from the default) which will have more boundaries as you would then control who will have access to that environment and on what level. Alternatively, you might inspect the possibility of creating an application user in SQL with read/write only to the table being used in the application – on the assumption that no filtering at user level is done at application level (which else would be bypassed from SQL Access).

PowerApps - Development

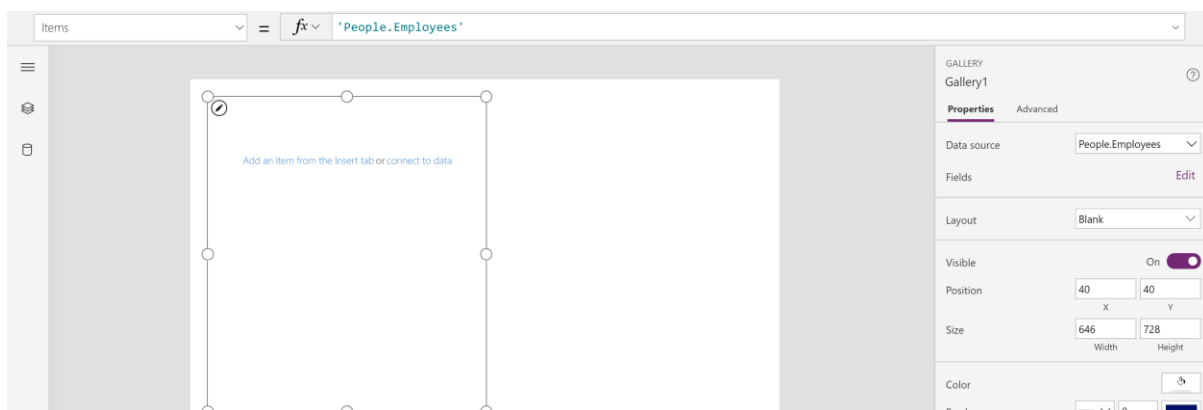
This section will be applicable to both the Excel and the SQL datasource.

We will now be doing an application that allows the list of the existing records in the People.Employees table, and also a way to edit or create new records.

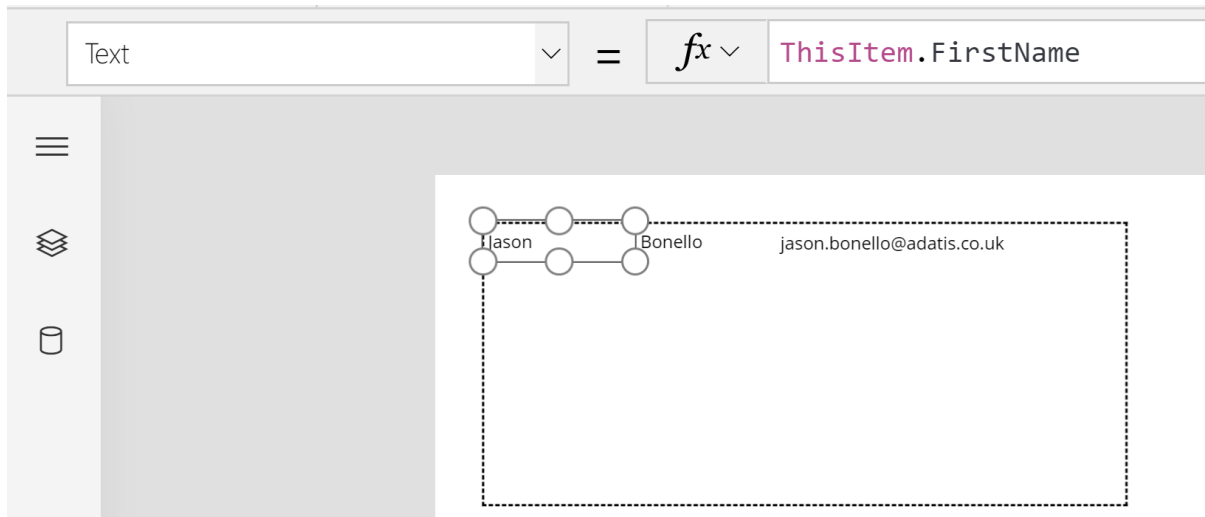
There might be different ways within PowerApps to do this. For this exercise we will be taking the following approach.

From the top ribbon, choose Insert, to have the controls listed.

From Gallery, choose Blank flexible height. The gallery is a predefined template in a form of a list, that allows the set-up of the initial row and replicates the setting for the rest of the rows in the list. As its data source, choose the table we have added in the connections.



With the Gallery selected, click on the top left pencil to be able to define the initial row. Then add 3 labels (again Insert > Label) and change the logic for each label in the Text attribute as “ThisItem.FirstName”, “ThisItem.LastName” and “ThisItem.Email” respectively.

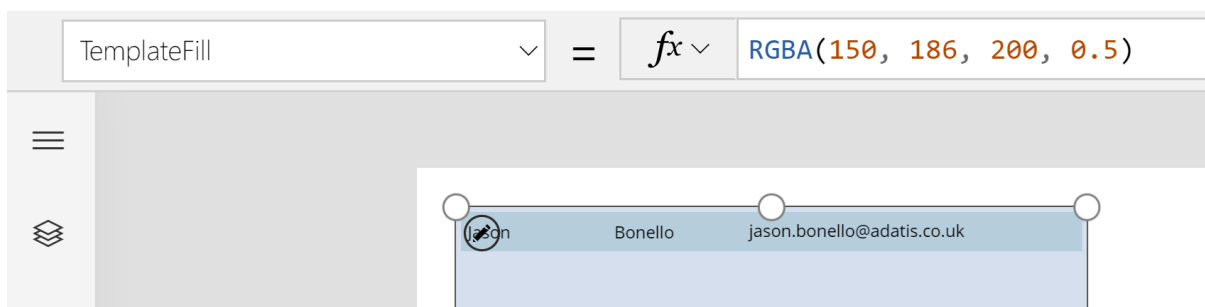


Align all three at the top of the box (select them all, then go Home>Align top), and re-size the first row (not the entire gallery) to take the height of these labels, as follows:



With the gallery selected, go to the Fill attribute (either the top bar or on the right hand-side of the screen), and choose a colour as a background for the gallery – I will be choosing RGBA(214, 223, 238, 1).

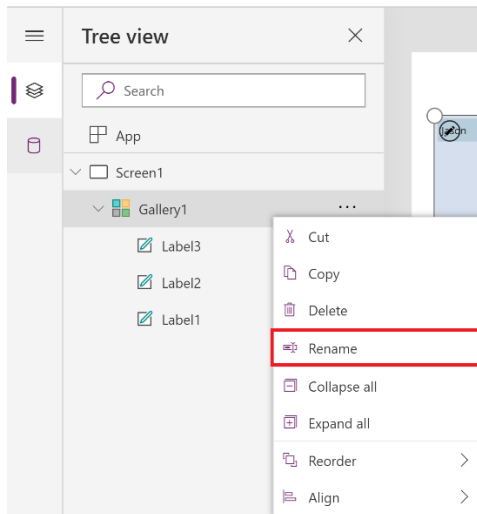
Then change the attribute of the gallery from to “Template Fill” and set it to a different colour, in my case I am going with RGBA(150, 186, 200, 0.5). This will look similar to the below:



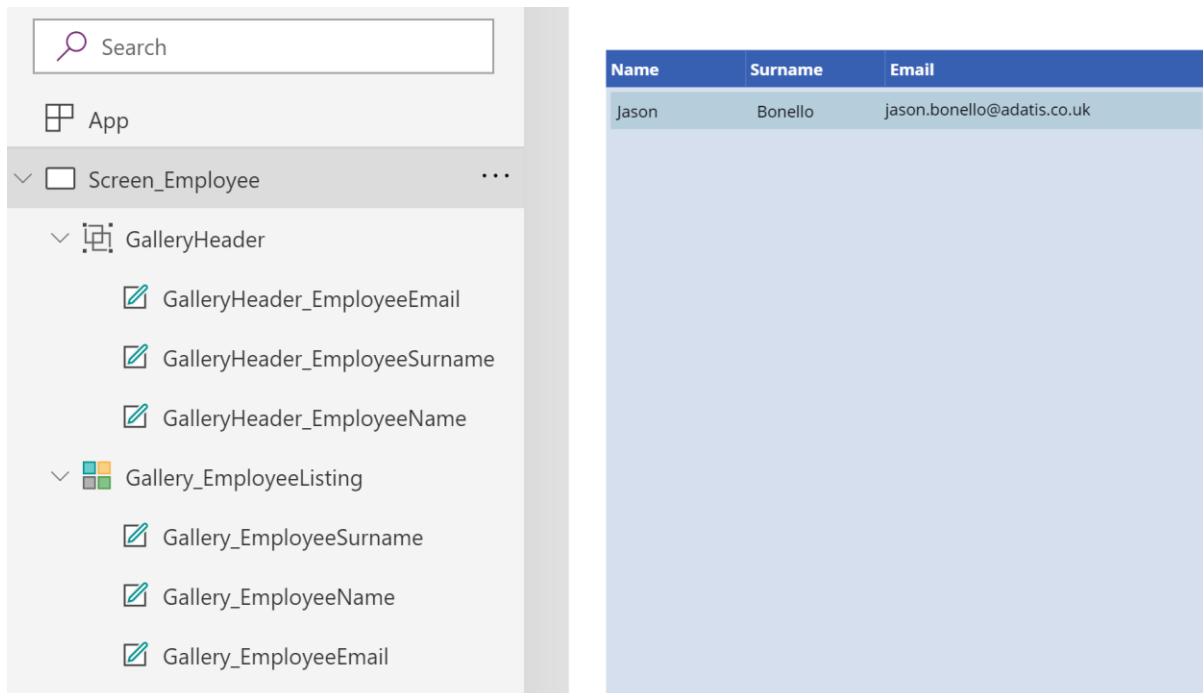
Outside the gallery (by clicking on any other part of the canvas outside the gallery itself), I will be adding 3 more labels, one as a header for each value – and replace the text of the label with “Name”, “Surname” and “Email”. These will be manually dragged on top of each field – and as a background I will be doing a different colour - RGBA(56, 96, 178, 1), and their font colour to white and bold. With the 3 or them selected, you can also right-click and group them in order to be able to move them all at once and treat them as one group.

As a best practice, always remember to rename the objects added on the canvas, as PowerApps will provide an auto-generated name, and whilst it will seem controllable when you are dealing with up to 10 objects, it may become messy.

This can be done by going on the Tree View from the left most vertical ribbon, and renaming all objects through a right-click on each.



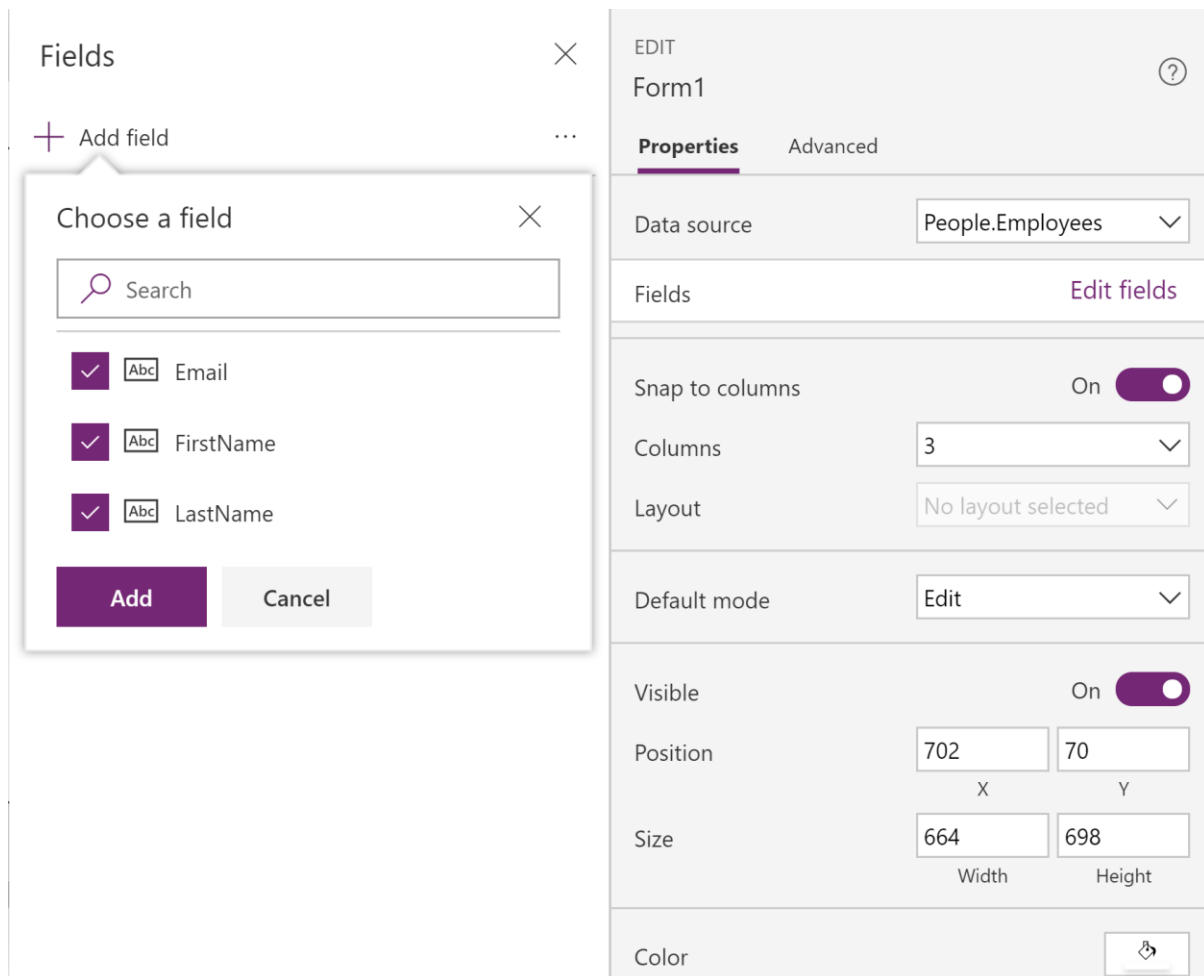
For this exercise, I will be referring to the already added objects in the following manner:



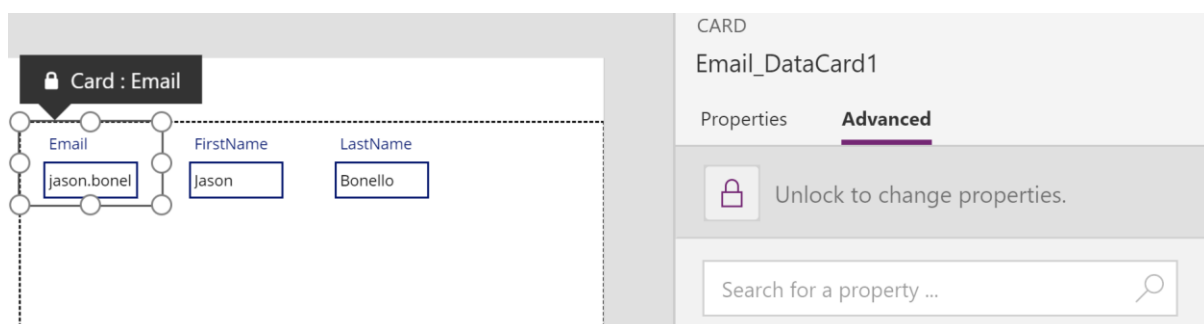
Now that the listing part is at an advanced state, let's start focusing also on getting the edit and creation screen ready. One easy way of doing this is by using the Form (Insert > Forms > Edit). Once added on the canvas, set its datasource to the same table, i.e. "People.Employees".

Apart from that, we also need to set the Item attribute for the Form, and we will be setting it as "Gallery_EmployeeListing.Selected". This would mean that anything that is selected in the Gallery, will have its details shown in the Form.

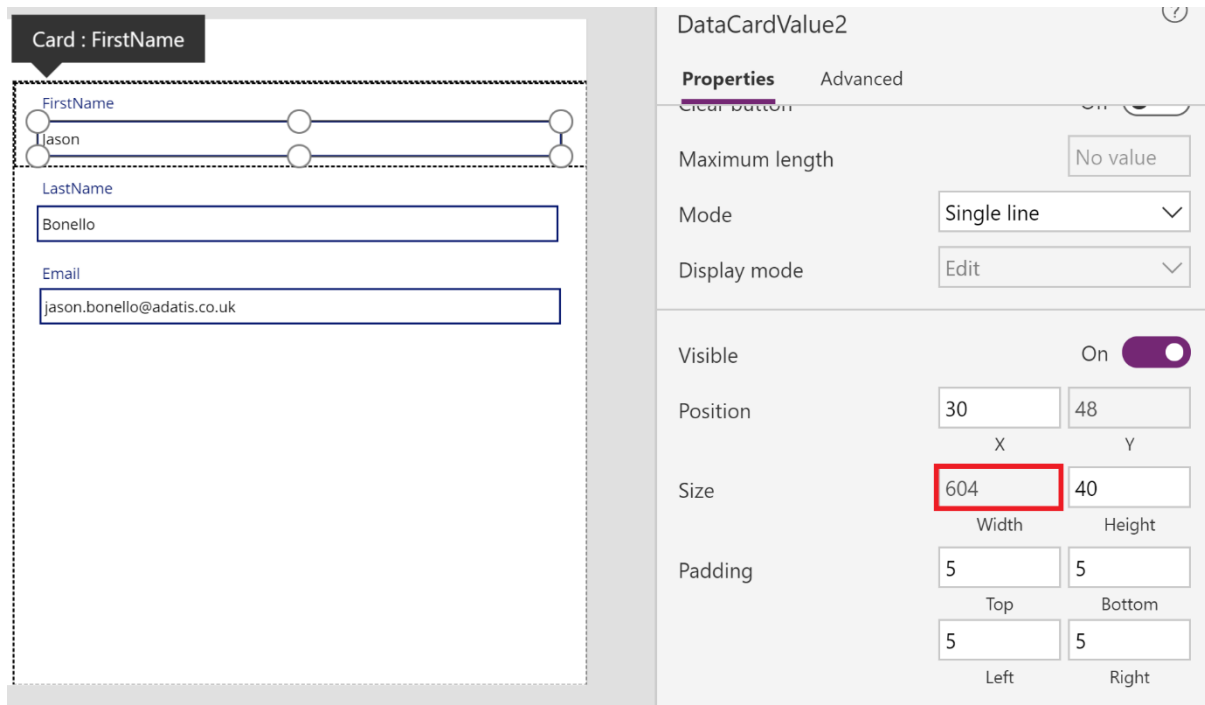
Now go to Edit Fields (whilst having the Form selected, just beneath the Data source in Properties on the far right).



Click Add and go to the canvas. Click on each data card, go to Advanced, and click on “Unlock to change properties”. This will allow re-sizing the object and other settings related to each datacard.



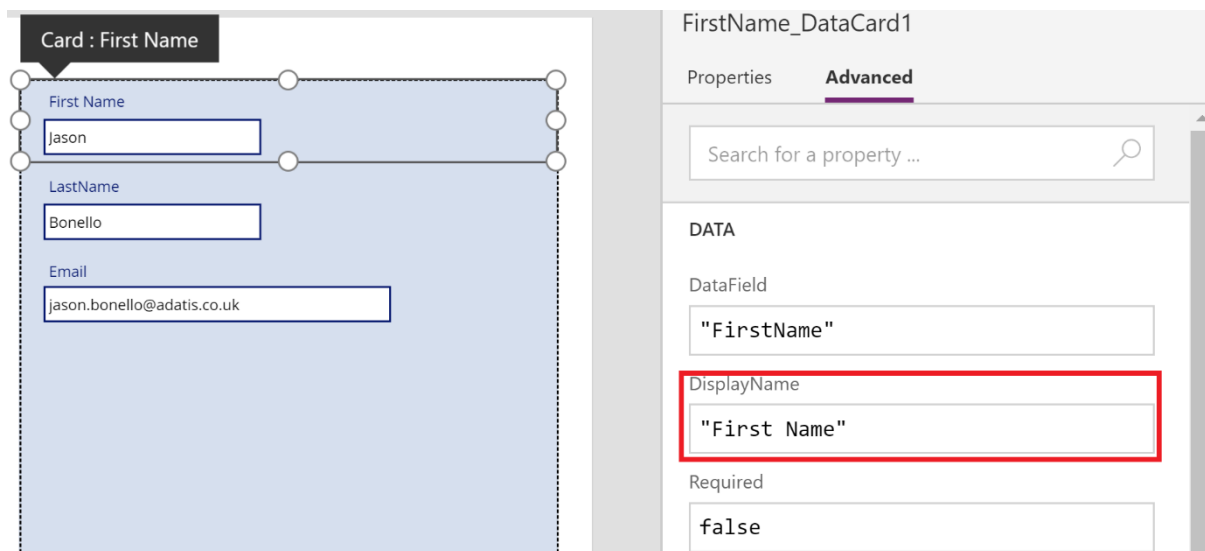
After doing so, you can drag the middle-right point to cover the entire Form space, and hence have one datacard per line.



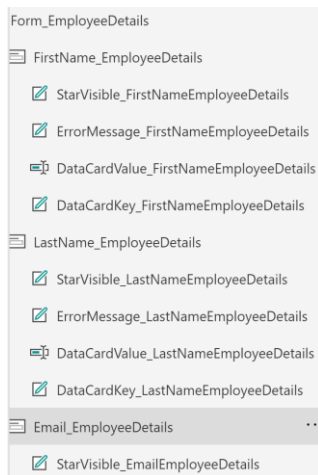
Then, by clicking on each datavalue field in each datacard, you could adjust the width of each field, to make it look better. For instance, we can set the Name and Surname datavalue fields to width 250 and the email to 400.

Change the background of the Form, in this case we are filling it to RGBA(214, 223, 238, 1) also.

Then, by clicking on each Datacard again, we will be able to change the Display Name to add a space in the Datacard Key (or field title).



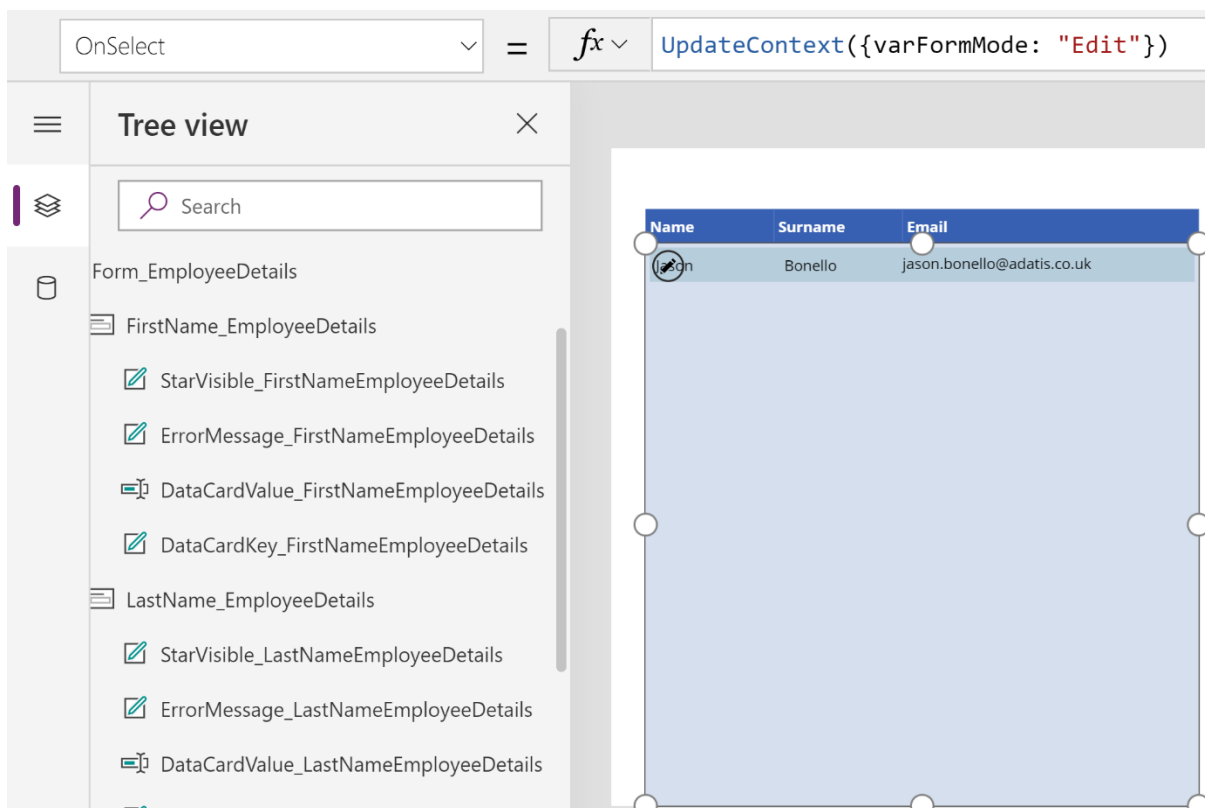
The objects for the Form were also renamed using the “_EmployeeDetails” suffix as per below:



So till now we are able to see the records as a list, and the selected record will show the details on the Form.

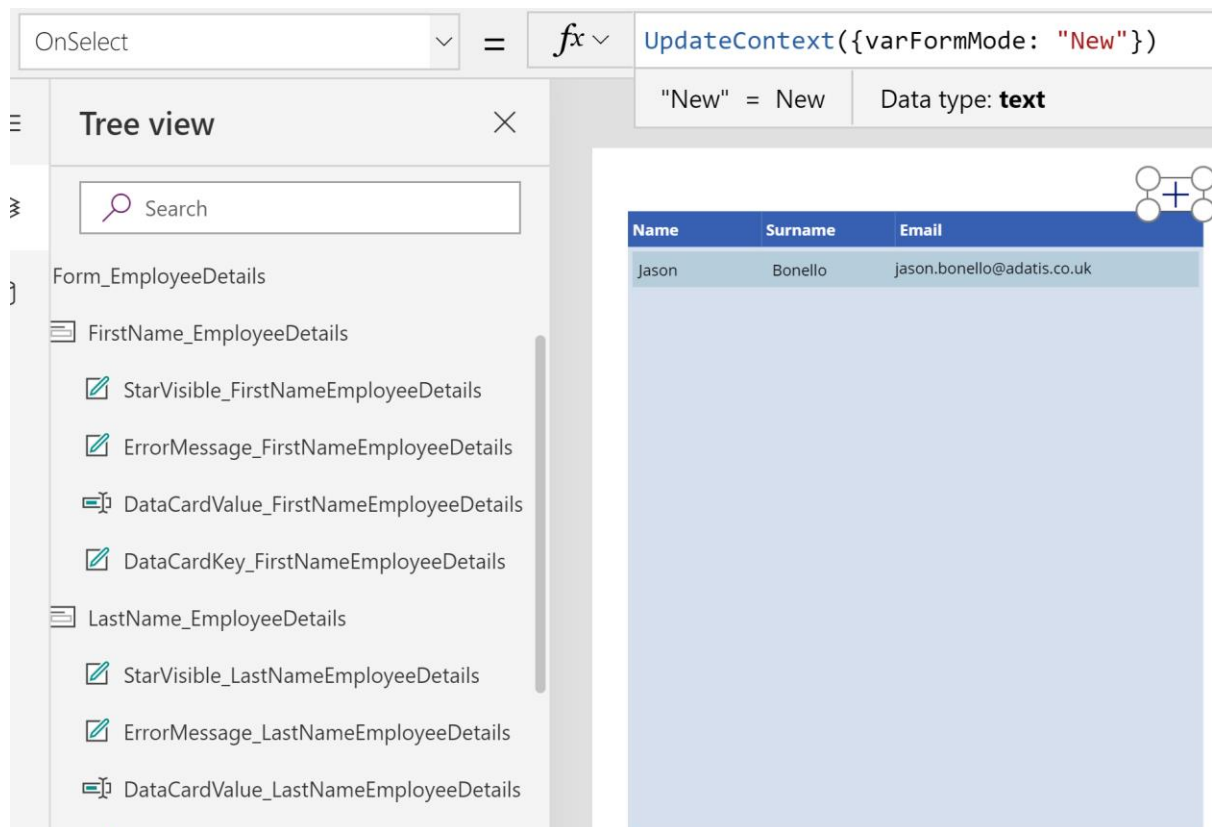
We now need to work on updating the data from the PowerApps into the source/destination table.

We are going to create a variable to act as a temporary placeholder to hold a value. As with other programming technologies, the variable can either locally in a screen or globally within the application – and what differs between both in PowerApps is the way we declare them. If we use `UpdateContext()` then that will be a local variable. If we use `Set()` then that will be a global one. In this case, we will be using `UpdateContext()`. With the Gallery List chosen, we will be setting the `OnSelect` attribute to `UpdateContext({varFormMode: "Edit"})`

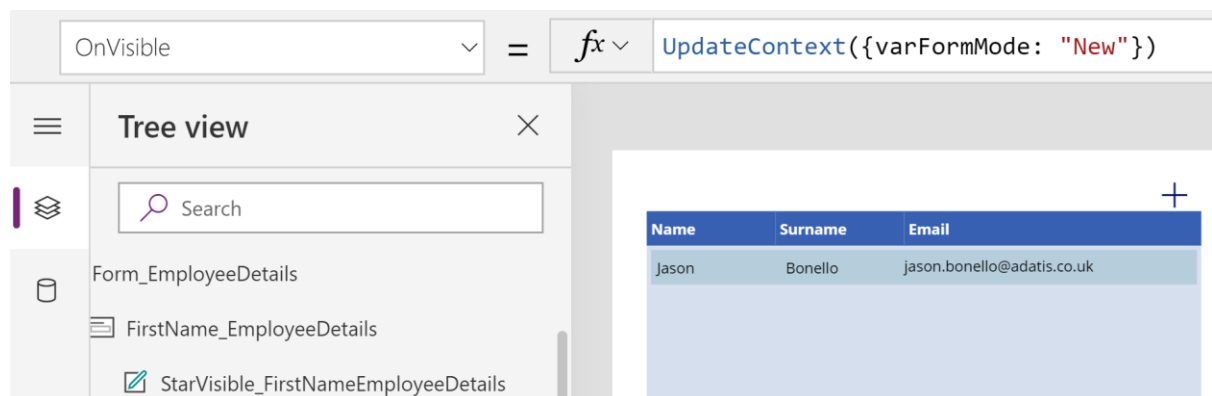


This way, once the user clicks on any of the values inside the Gallery, the variable named “varFormMode” will automatically be set to “Edit”. Now we are going to add an Icon (Insert > Icons)

and choose the “+” which is the add sign, dragging it on the top right of the gallery header ribbon in the canvas. The OnSelect of the icon, is to be set as `UpdateContext({varFormMode: "New"})`



To also ensure this variable is properly handled in all cases, we can press anywhere in the canvas where an object does not exist and set the variable to “New” by default in the OnVisible attribute.



This means that as soon as this screen is to become visible, the variable is going to be initiated with the “New” value;

We are now going to add a Check icon from the Icons (Insert > Icons) and place it on the top right of the Form. We will be using this to determine the update of an existing or creation of a new record. And in the OnSelect we will place the following logic:

```
If(
    varFormMode = "Edit",
    Patch(
        'People.Employees',
```

```

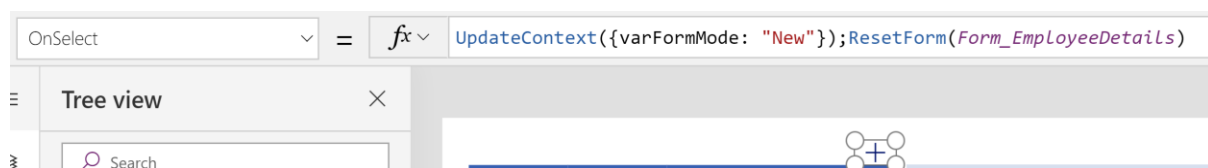
    Gallery_EmployeeListing.Selected,
    {
        FirstName: DataCardValue_FirstNameEmployeeDetails.Text,
        LastName: DataCardValue_LastNameEmployeeDetails.Text,
        Email: DataCardValue_EmailEmployeeDetails.Text
    }
),
Patch(
    'People.Employees',
    Defaults('People.Employees'),
    {
        FirstName: DataCardValue_FirstNameEmployeeDetails.Text,
        LastName: DataCardValue_LastNameEmployeeDetails.Text,
        Email: DataCardValue_EmailEmployeeDetails.Text
    }
)
)

```

The patch function in PowerApps will update or create a record in the connected table - see further details on it here: <https://docs.microsoft.com/en-us/powerapps/maker/canvas-apps/functions/function-patch>

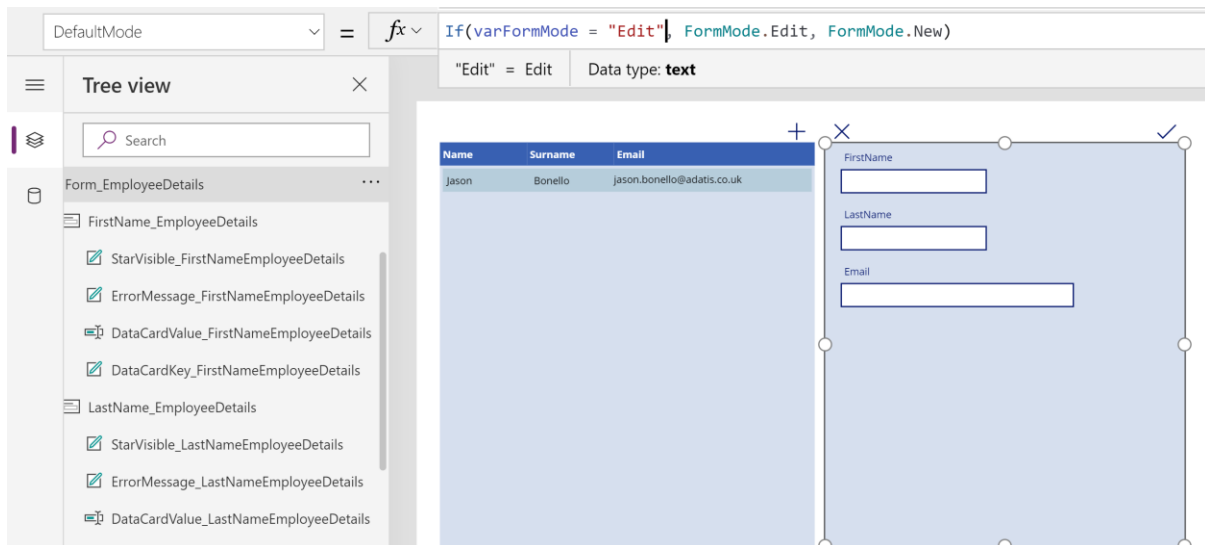
What we are doing in our added logic, is just an IF condition, to check what the value of the varFormMode is. And if it is “Edit” then we are doing an update of the Selected record within the table. Else we are creating a new record.

Now in the interface we still need to take care of clearing the form if we click on the “+” sign. So in its OnSelect, we can add another command ResetForm(Form_EmployeeDetails) and separate the existing command and this with a “;” – which is what allows multiple functions to take place through one trigger in PowerApps. The ResetForm on the other hand is self-explanatory, and will clear out the fields of the form.



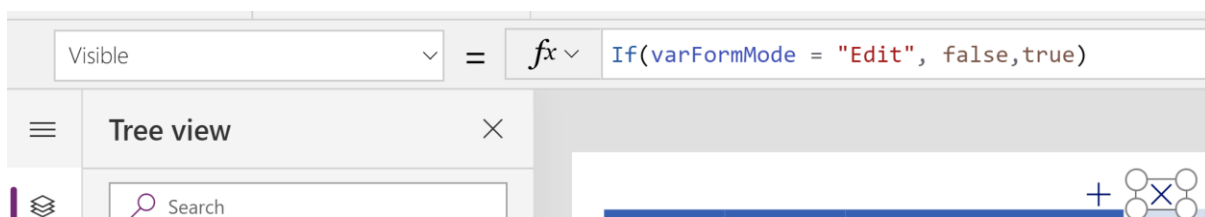
We can likewise add the same to the OnVisible of the canvas screen (so that on loading the screen we are sure that the form is reset and cleared).

An important step to keep in mind when dealing with Forms is the “Default Mode” attribute. This determines whether the Form will be in a state of editing an existing record, in a state of viewing (without changes allowed) or in a state of creating a new record. So with the whole Form selected, we need to add If(varFormMode = “Edit”, FormMode.Edit, FormMode.New)



Hence the varFormMode will be determining whether we are setting the Form to be in edit state or in a new state.

We can also add another icon – “X” to be able to clear any information to be inputted when creating a new record. In its OnSelect attribute we will be placing ResetForm(Form_EmployeeDetails) and also in its Visible attribute, we will put If(varFormMode = “Edit”, false,true)



This means that the “X” icon will only be visible if the varFormMode is “New” and else it will be hidden. When visible, when clicked, it will Reset the form (in case a user needs to clear all fields and cancel posting).

At this point we can also add 2 rectangles, send to the back of the icons, change the icons’ colour to white and also add a label as a title.

The app will look something like this:

Employee Details

Name	Surname	Email
Jason	Bonello	jason.bonello@adatis.co.uk

+

×

✓

FirstName

LastName

Email

Even if it is advised to this frequently along the course of development, we will be saving the application to the environment. Saving the application only, would not make the changes accessible and visible to the users. Publishing is what is required to make the application a useable release. Hence always remember to publish the application once you are happy and complete with the developments.

We can use the “Share this app” shortcut to provide access to other users.

EmployeeApp_Excel

Environment: N/A (default)

Saved: 06/10/2019, 18:14:28

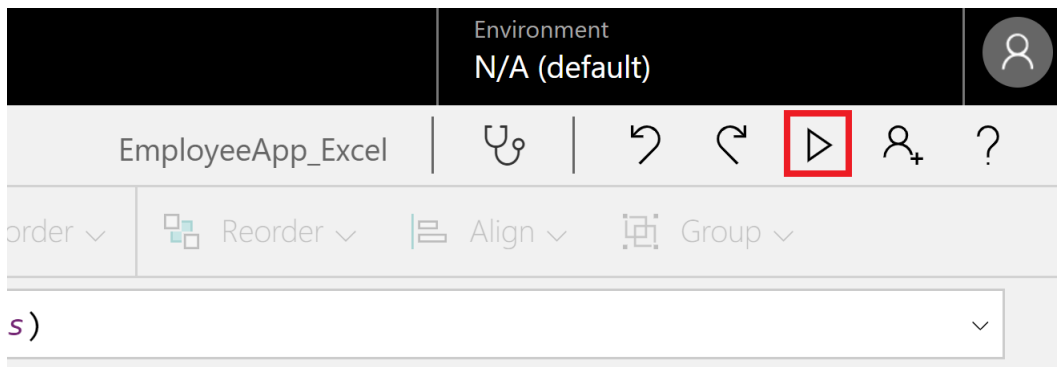


All changes are saved and published.

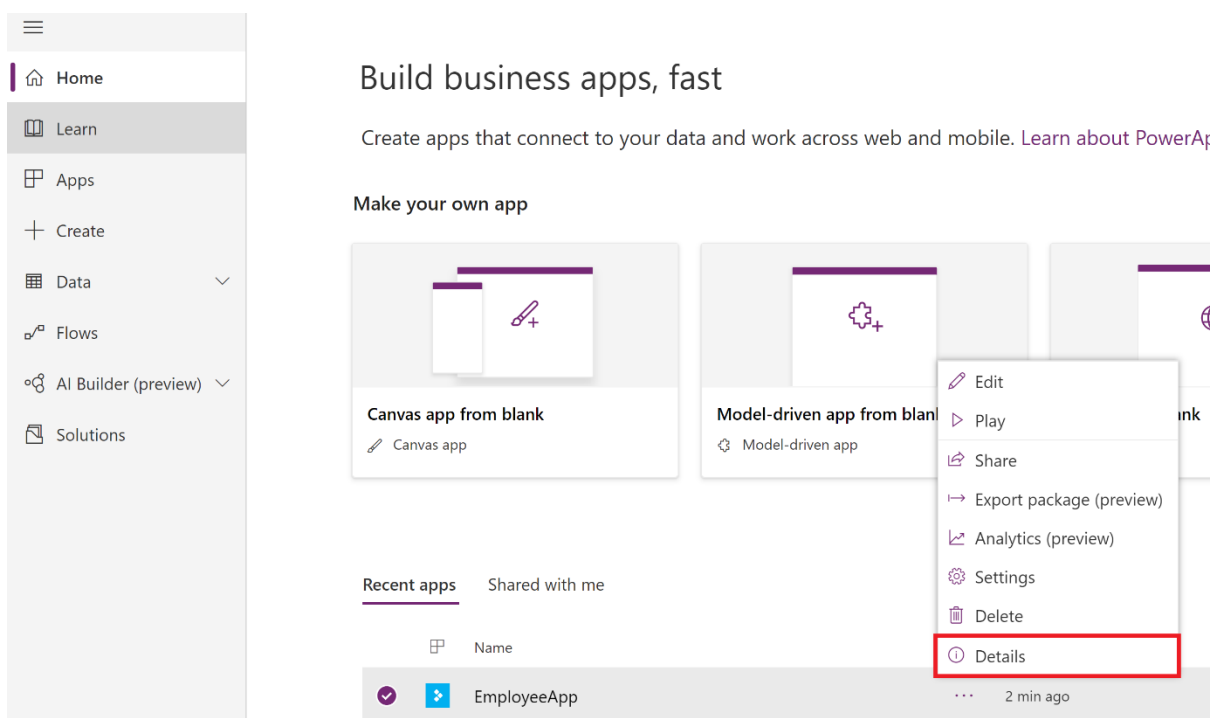
[Share this app](#)

[See all versions](#)

To play the application in the same way the user is going to access the published version of it, use the “Play” icon.



Else, once published, users with access to it, can access it through the generated App URL. The URL can be seen though the Apps screen upon logging in PowerApps, by clicking on the three dots icon of that particular application and going into “Details” – as per below:



Thanks for following – if you have any questions reach me out on any of the following.

Email: jason.bonello@adatis.co.uk

Twitter: @bonello_jason

Blog: <https://adatis.co.uk/author/jasonbonello/>

