Power Platform

Model-Driven Applications for Application Users

Who is this training for? 3

What are we going to look at 5

Basic Navigation in the Model-Driven Applications 6

Model-Driven Applications and Shared Data 15

Working with views 17

Personalization Settings 22

Working with the data 26

Forms 34

History tracking and audit capabilities 50

Coming soon 52

Live Course Delivery 52

Learn it today,
start using right
away

Power Platform learning course by Alex Shlega (MVP)

1. Who is this training for?

It is a valid question, since this training is part of the bigger “Power Platform training course” which is meant for the application makers / developers. However, the purpose of this particular module is to provide training in the core capabilities of Model-Driven applications to both end users and application makers, especially if the latter are just starting to work with Power Platform. To clarify this a little more, and if you are not too familiar with the model-driven applications yet, I’ll resort to an analogy.

Imagine that you’ve been given a jet plane, and you were asked to fly from point A to point B (Those of you who have happened to have a jet pilot licence may have to also imagine you had never had it).

You know that jet plane can get the job done, and you may even have some idea of how it’s supposed to be done, but, with all likelihood, if you try without prior guidance, you might end up ruining the jet plane, the surroundings, and, potentially, you might be risking your life.

It is not too far from what’s happening on some Power Platform / Dynamics projects. You get a business analyst or a UI designer who has limited understanding of the platform capabilities, and they start coming up with the requirements and/or requests which are just not possible to implement, so the project gets ruined.

Or there are users who do not understand how to work with the model-driven applications functionality which is there out of the box, and they start complaining that certain functionality is not available. The project starts receiving bad feedback, and, eventually, gets ruined.

Or, possibly, a developer creating a model-driven application does not realize what a business process flow is, and, out of a sudden, your project ends up being way over budget since there is a custom implementation of the process stages.Or, possibly, a user reporting a bug might not be providing enough details for some reason, so a developer looking at the same feature in the dev environment has to be able to guess what is it the user was doing.

Unlike with the Canvas Applications, where the applications are developed from the ground up, in the model-driven applications world we often only need to define which pieces of the data model should be available in that application.

But, once it is done, how do we know what other functionality will be provided by the platform so we would not have to implement it, and so that the users could rely on it?

This is where this guide comes in, and, hopefully, it’ll be useful to both the end users and the developers.

2. What are we going to look at

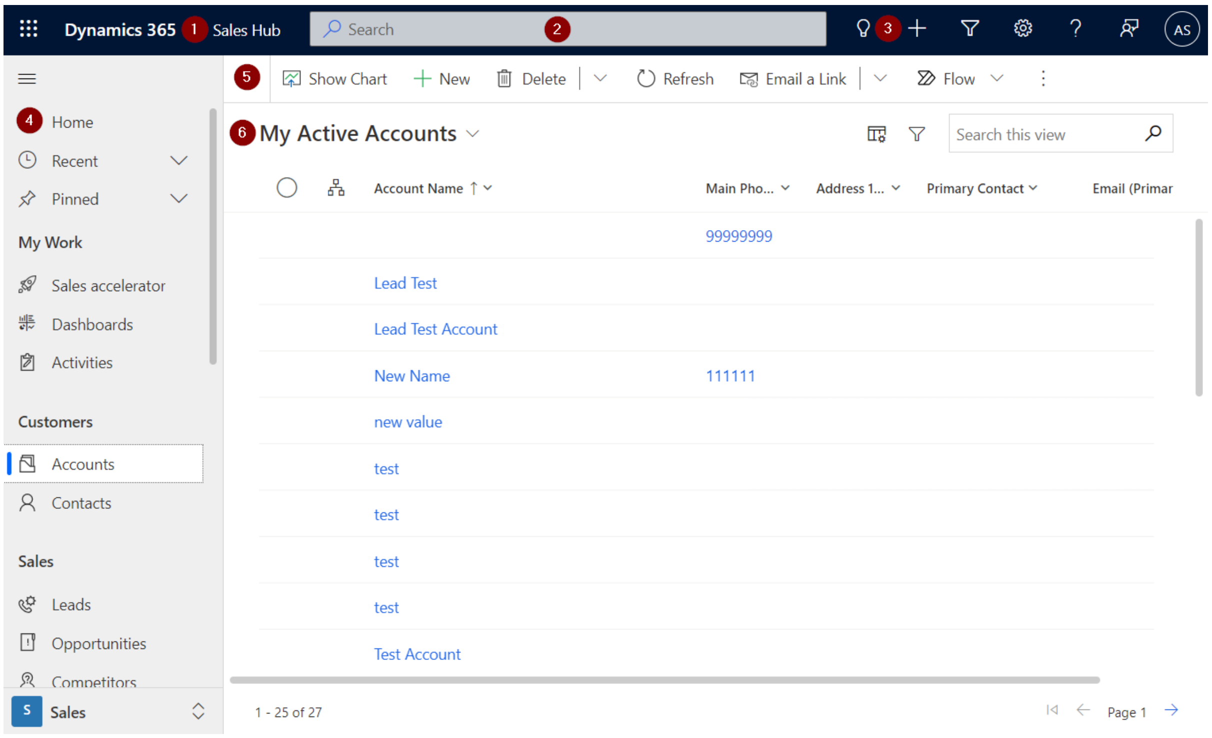
Here is what we are going to look at:

- Basic navigation in the Model-Driven Applications
- Model-Driven Applications and Shared Data
- Working with views
- Personalization options
- Working with records
- Forms navigation (sections, tabs, related, etc)
- Command Bars
- Charts
- Dashboards
- Search and reporting options
- Sharepoint integration, Email integration, Word Templates

Keep in mind that, in this module only, we are going to look at those features more from the user perspective than from the configuration and customization standpoint. The idea is not to learn how to set it up, but, instead, to see how it works and to get basic understanding of the out-of-the-box features (even if some of them may need to be configured by the system administrator before application users can start using them)

3. Basic Navigation in the Model-Driven Applications

All Model-Driven applications offer the same navigation framework:



Account Name ↑	Main Pho...	Address 1...	Primary Contact	Email (Primar
	99999999			
Lead Test				
Lead Test Account				
New Name	111111			
new value				
test				
test				
test				
test				
Test Account				

There is application selector (1), which is there for any model-driven application

There is search box (2), which is there all the time as well

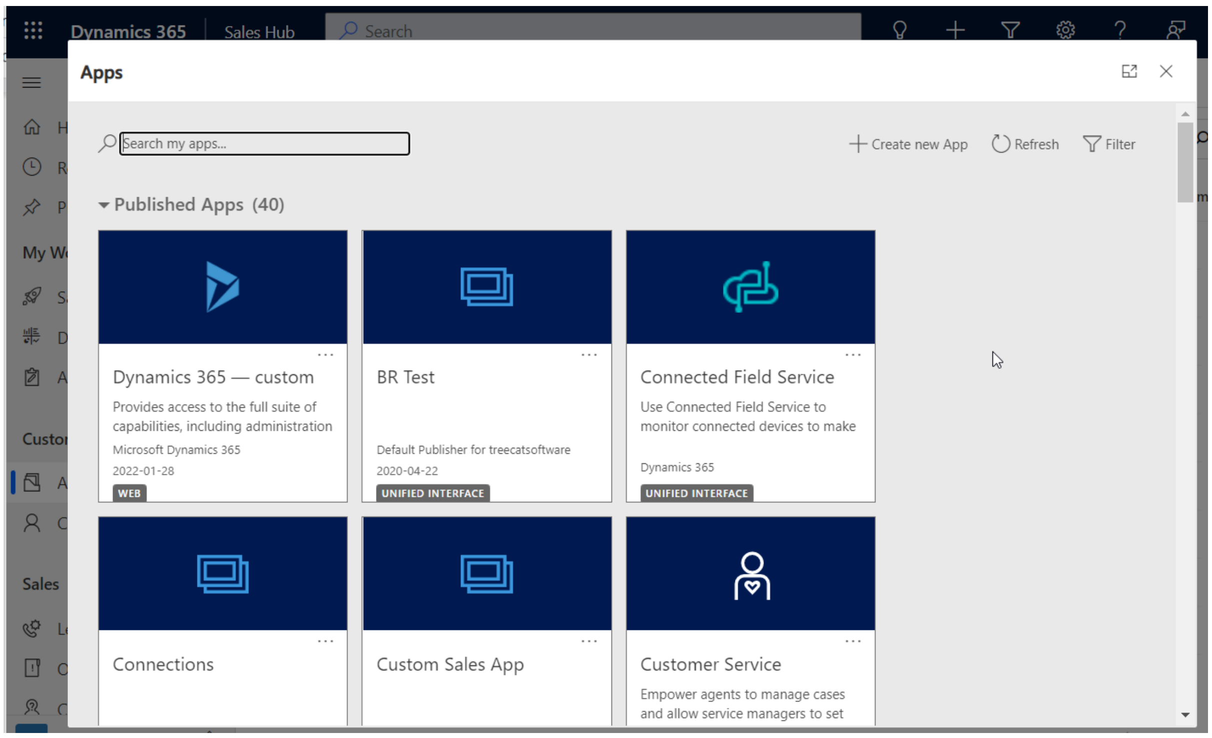
There is quick access menu (3), which does not change from app to app

There is navigation area (4), which is app-specific

There is command bar (5) which is context-based

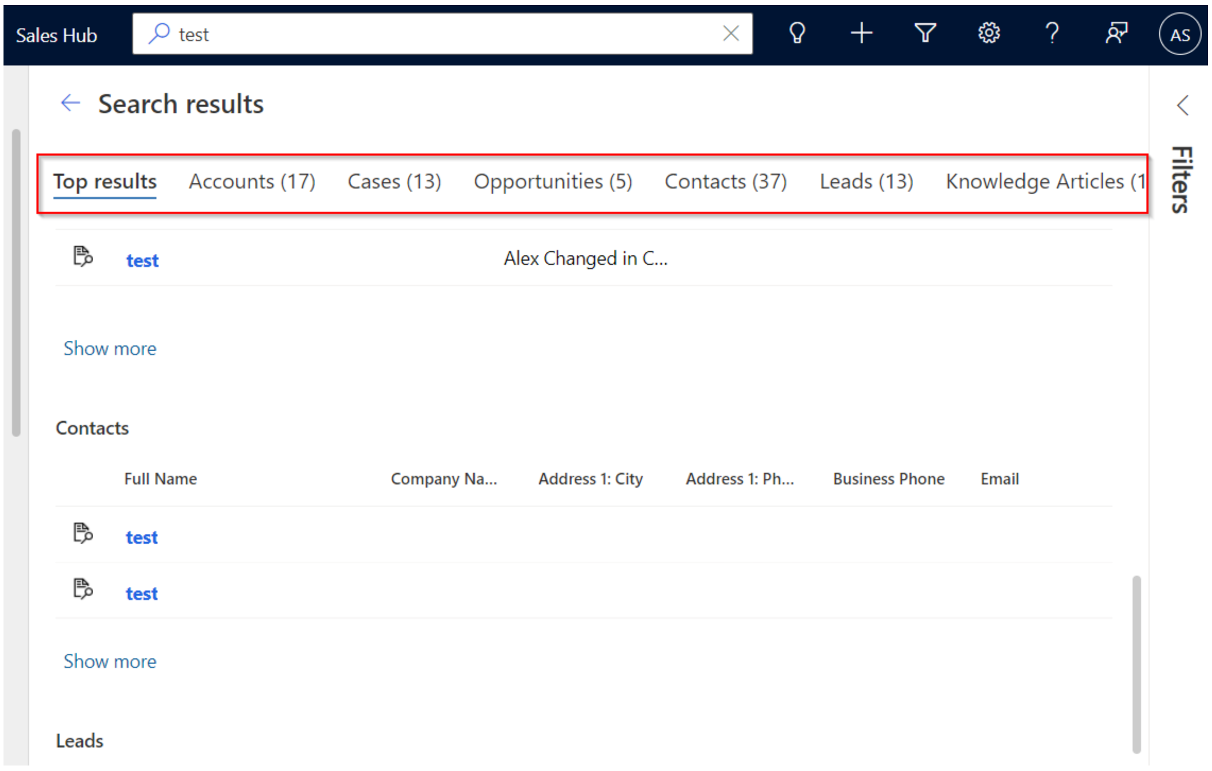
And there is content area (6)

While working in the application, you can click application name (“Sales Hub” on the screenshot above) and pick another application from the list of apps which you have been given access to:



If you are a system administrator in the environment, you will have access to all the apps there. Otherwise, depending on the security roles assigned to them, different users will see different applications.

The search box offers some surprisingly powerful functionality - keep in mind it has to be configured properly, but, once it’s been configured, you can use it to search across multiple tables at the same time::



test Alex Changed in C...

Show more

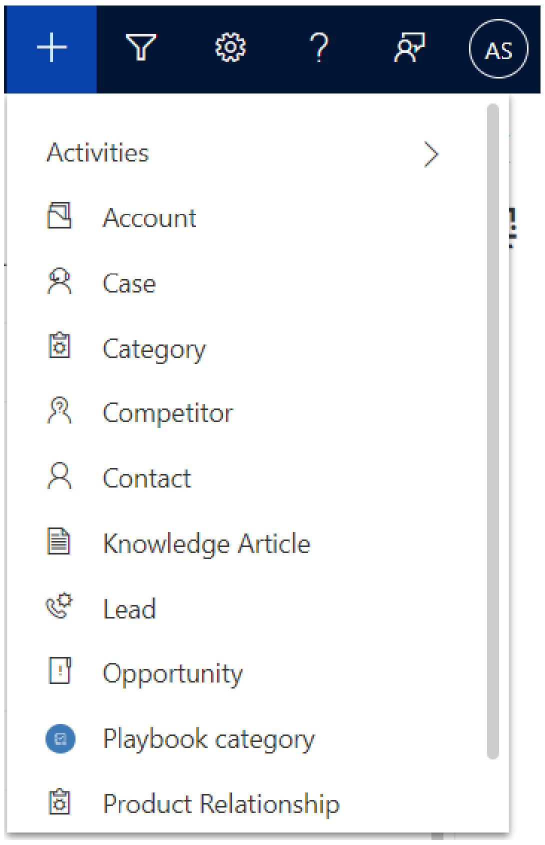
Contacts

Full Name	Company Na...	Address 1: City	Address 1: Ph...	Business Phone	Email
test					
test					

Show more

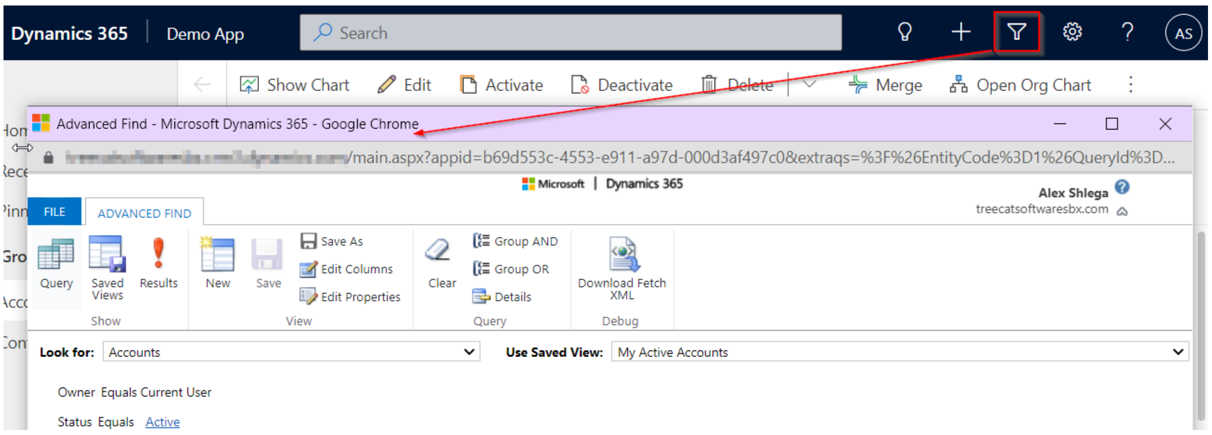
Leads

There are a few useful options in the quick access menu. For instance, you can use it to start creating records quickly:



Not all the tables will show up there - those tables have to be configured so that “quick create” is enabled for them, and, also, they have to be added to the model-driven application.

You can also use that menu to quickly access “Advanced Find”:



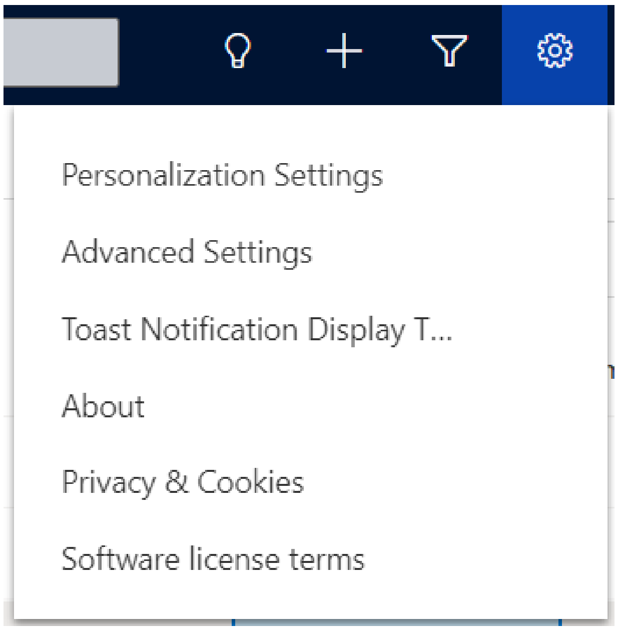
Look for: Accounts
Use Saved View: My Active Accounts

Owner Equals Current User

Status Equals Active

Advanced Find is, essentially, a way to build custom queries. You can use it to search for the records from a specific table based on the criteria you'll define in the advanced find window. We are not going to discuss it in detail right now, but we'll get there a bit later in this module.

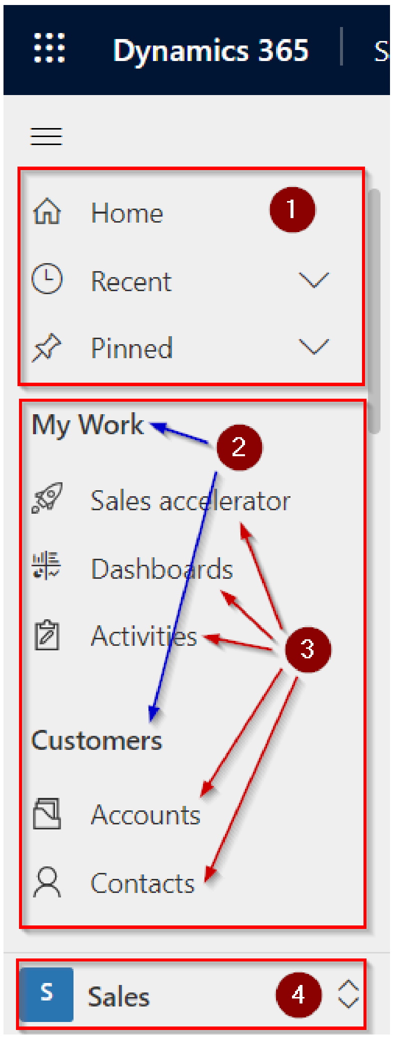
Finally, there is “Settings” button:



We will look at the Personalization Settings later in the module, but, in short, this is where you can update personal settings such as the number of records you want to be displayed per page, for instance.

Advanced Settings are more applicable to what System Administrators will be doing in the system. And we don't need to be concerned with the other options yet.

Then there is application navigation area:



Every model-driven application will expose its own navigation items on the left side, so that there will be a few areas (4), each organized into groups (2) and subareas (3).

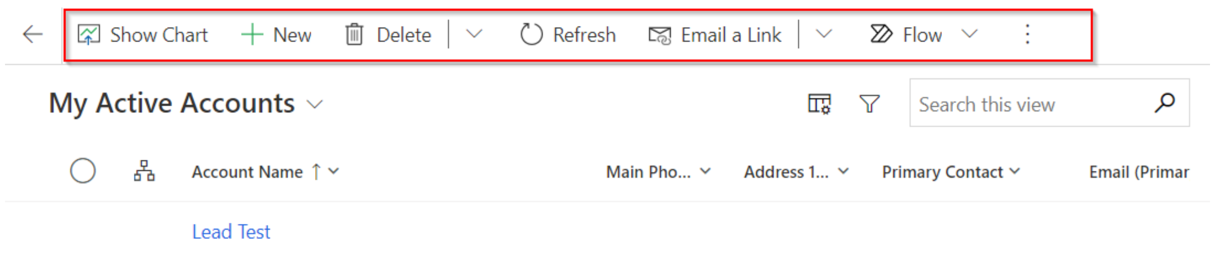
It's the subareas which will represent actual navigation items in the application. More often than not, each subarea will correspond to a table, but it can also be a dashboard, an HTML page, or, possibly, an external link.

Only one area will be visible at any time, but you will be able to select other areas by clicking on the area selector next to the area name (up and down arrows next to the "#4" on the screenshot). And groups (2) are there just to help with the grouping of different subareas.

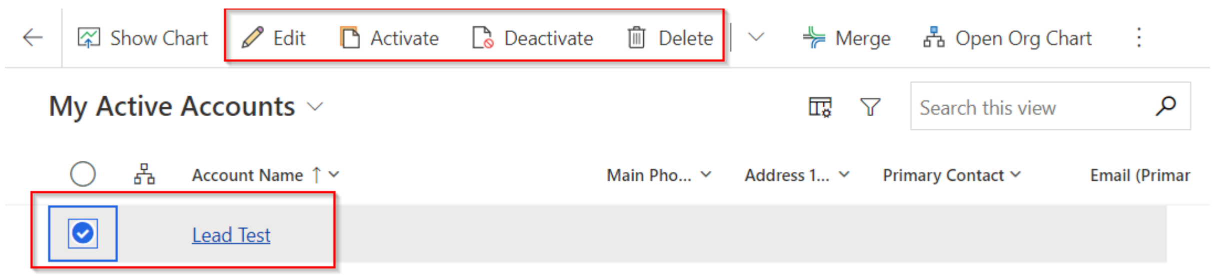
There is, also, common area (1) which is the same for all model-driven applications. This is where you can see pinned items, recent items, and, also, quickly jump to the "home" page.

You will notice **command bars** in different places in the model-driven applications. Those are areas on the screen filled with the command buttons such as “New”, “Delete”, “Refresh”, etc.

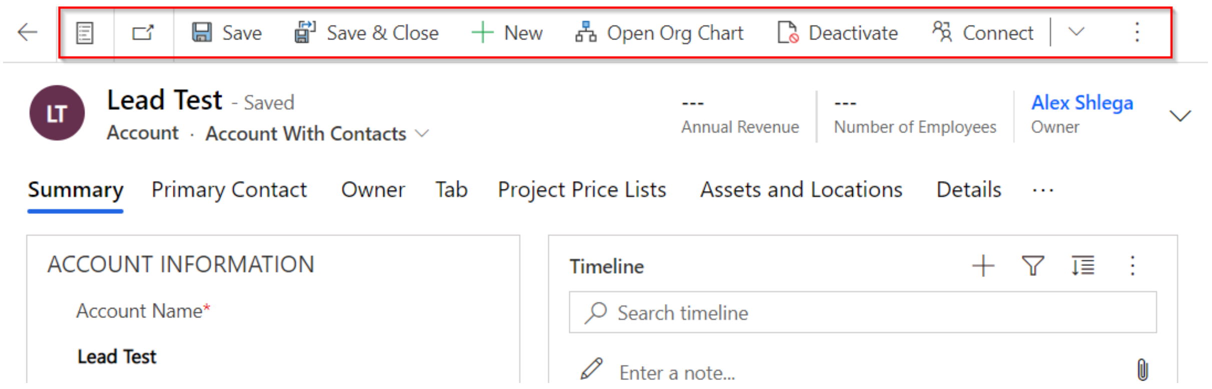
For example, when looking at the list of records, you will see a command bar which might look similar to the one below:



Keep in mind that command bars are contextual. In the example above, once a record has been selected, some additional button will show which only make sense in the context of the selected record:

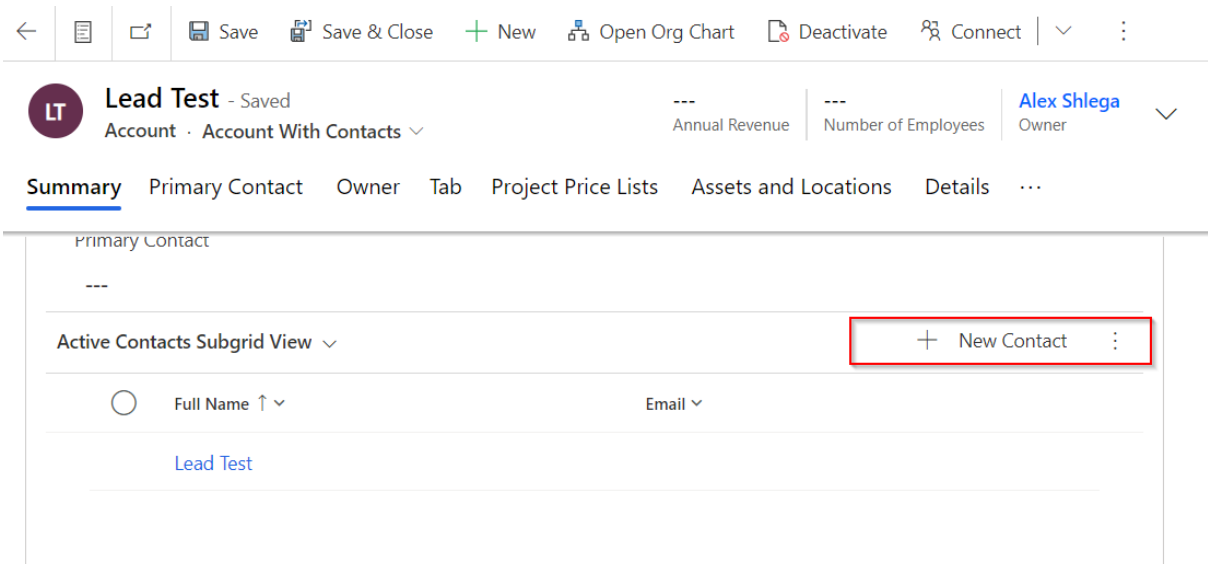


You will also see record-specific command bar once you have navigated to that record:

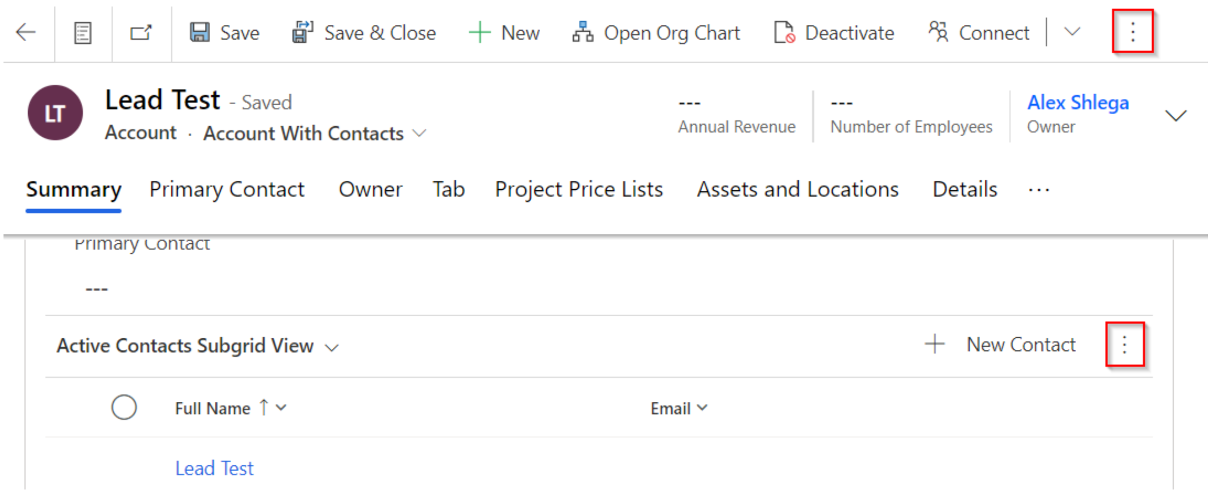


Notice how in the above example there is no "Show Chart" button, since that button only makes sense in the context of the list of records.

Model-drive applications will also display a command bar above every subgrid. In the example below, you can see two command bars. The first one is identical to the one you've seen above, but there is yet another one which is working in the context of the contacts associated to the account:



Keep in mind that, quite often, there will be additional command hidden under the "three dots" for each command bar:



In short, command bars are filled with various action buttons, which, in turn, are displayed in the context of the data currently displayed on the screen.

Developers have no control of where and how command bars are going to show up, but they do have some control of the buttons which are showing up there.

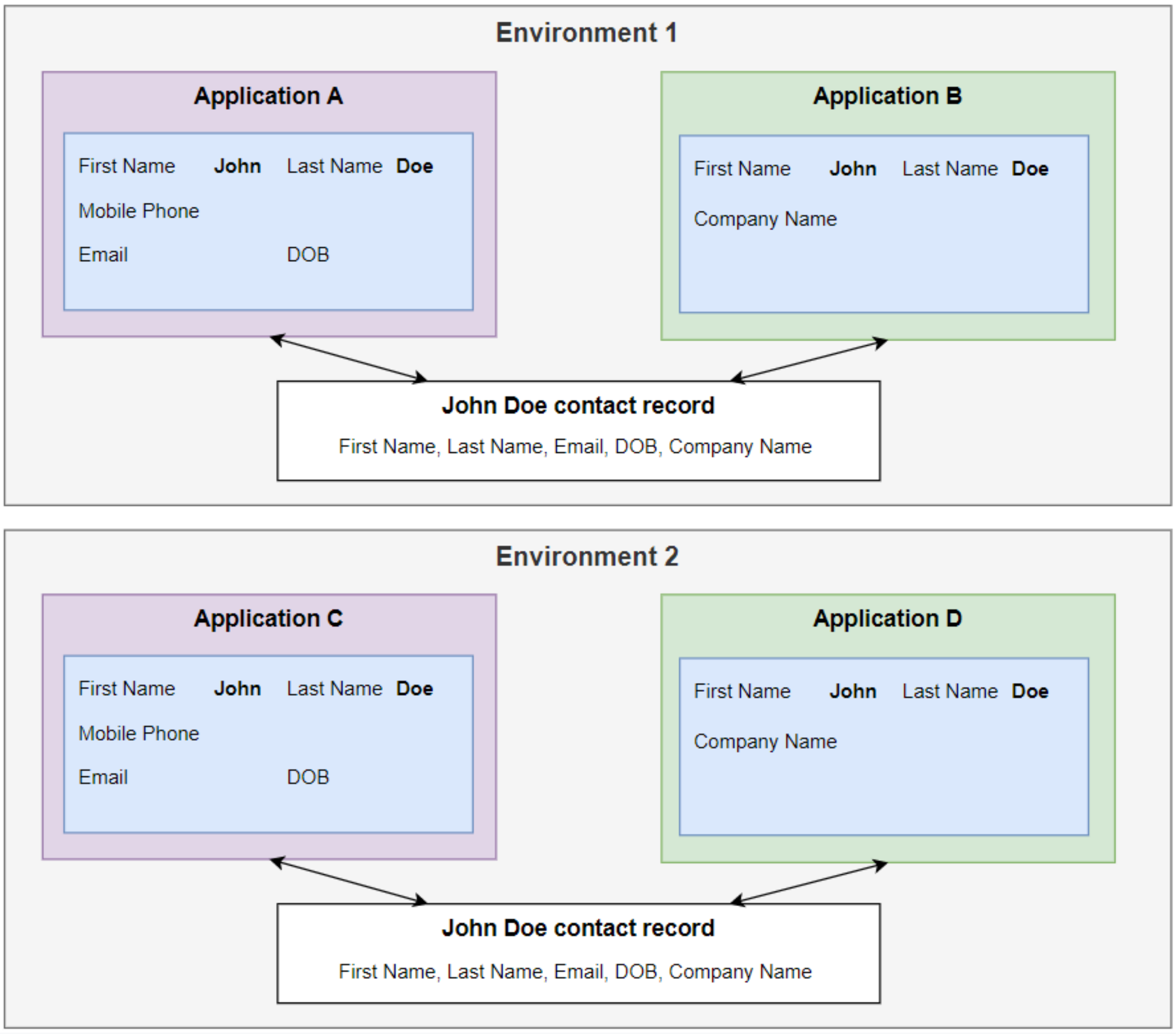
As for the **content area** in the middle of the screen, that is where model-driven applications will be displaying dashboards, lists of records, individual records, and, possibly, other content as you keep navigation through the application.

With all that said, let's clarify a few things:

- In every environment, there can be multiple model-driven applications. Depending on the permissions assigned to them, different users will see different applications
- Each application will define its own navigation structure
- From the security standpoint, access to the certain areas of the application may be further restricted by not assigned required security roles to the users. In other words, the same application may look somewhat different to different users

4. Model-Driven Applications and Shared Data

When thinking about model-driven applications, keep in mind that, quite often, the data will be shared between those applications.


Environment 1
Application A
First Name John Last Name Doe
Mobile Phone
Email DOB
Application B
First Name John Last Name Doe
Company Name
John Doe contact record
First Name, Last Name, Email, DOB, Company Name
Environment 2
Application C
First Name John Last Name Doe
Mobile Phone
Email DOB
Application D
First Name John Last Name Doe
Company Name
John Doe contact record
First Name, Last Name, Email, DOB, Company Name

What the screenshot above is meant to illustrate is that:

- There could be multiple environments in the same tenant
- Each environment can have multiple applications

- Each of those applications in the specific environment may expose pieces of the same data, and, when that data is updated in one application, those changes will surface in the other
- Just to clarify, though, model-driven applications cannot cross the boundaries of their respective environments when accessing data (at least not unless developers put significant effort into making it work between environments)

In other words, every application in a particular environment may look unique, but all of them may still be using the same data.

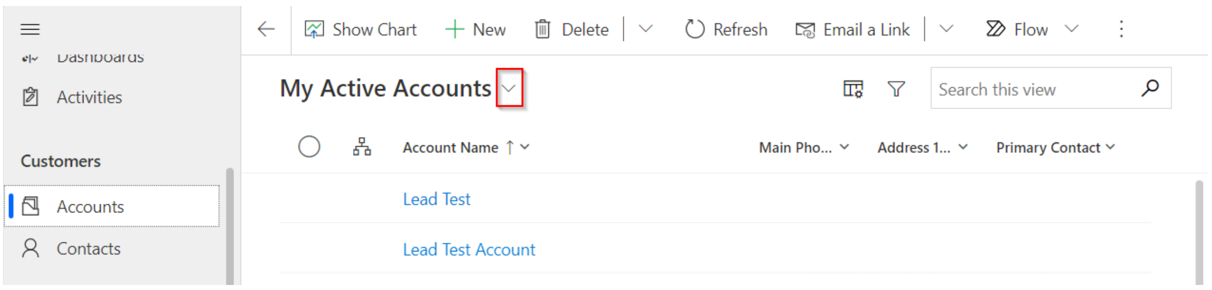
So, then, what if we need to isolate data between different applications? That’s doable, of course, but that’s a question of setting up a proper security model, or, possibly creating custom tables so that each application exposes its own table instead of using the same shared ones. That's something applications makers / developers would have to take care about when creating the application.

5. Working with views

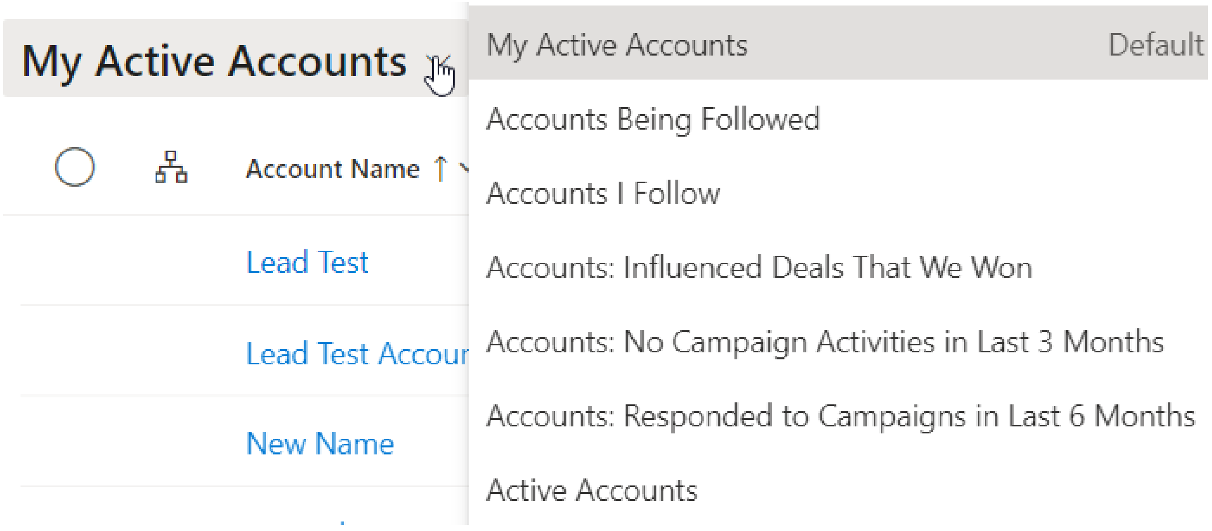
Views, Lists, or Grids - in the model-driven applications these are often referring to the same thing, though each of those words has a slightly different connotation.

A view in model driven applications is a combination of a query, columns, and sorting order. If you wanted to see a list of records from a certain table (or from that table and from related tables which are related to the main one through what is called 1:N relationships), you could use a pre-configured view, or you could even create a new one using Advanced Find.

For example, on the screenshot below you can see how “My Active Accounts” view is, currently, selected:



There is a dropdown icon next to it which provides access to a selection of other account views:



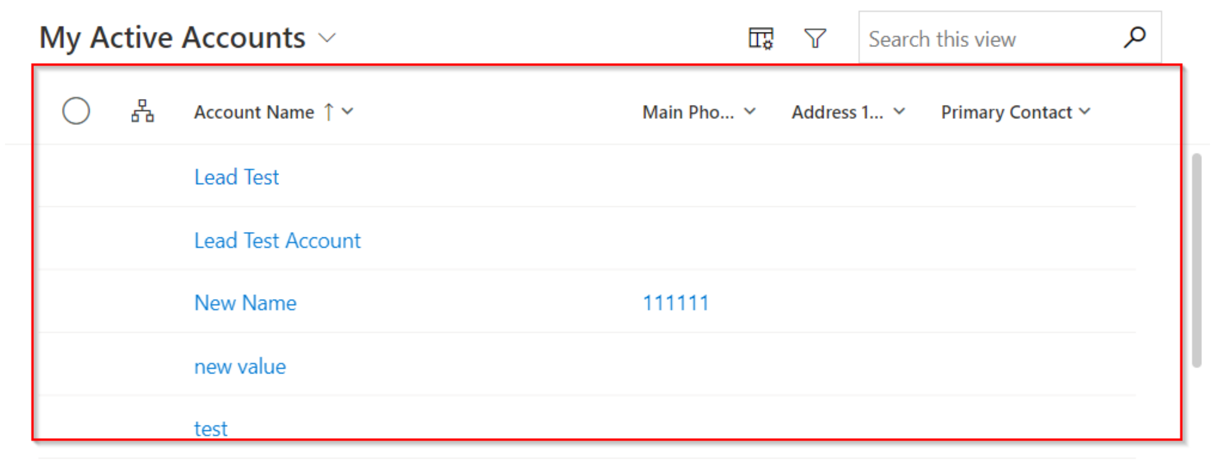
Some views are configured by the makers / developers, those are system views. Each application in the environment may choose to expose all or only some of those system views.

Users, on the other hand, have the ability to create their own views AND to share those views with other users or teams in the same environment.

To reiterate, here is what we can say about the views (among other things):

- They are named pre-configured queries
- Every view defines required query conditions, default sorting, and, also, list of columns that will be displayed when the view is selected
- There can be system views and personal (user) views
- Every application controls which system views it wants to expose
- Users can share their personal views with other users (unless that permission has been taken away)

Views are selectable in different places in the model-driven application, and, once a view is selected, a list of records will be rendered. Usually, that list will be rendered are a grid (or, possible, a subgrid, if that grid is displayed on another record's form). For instance, here is an example of such grid:

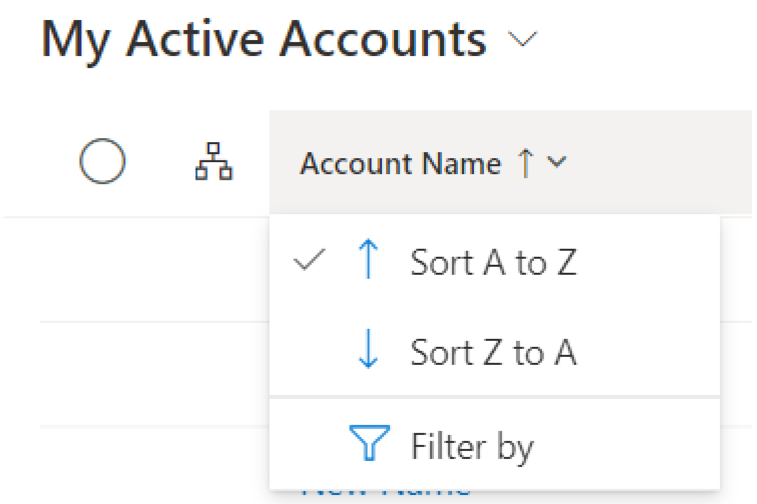


My Active Accounts

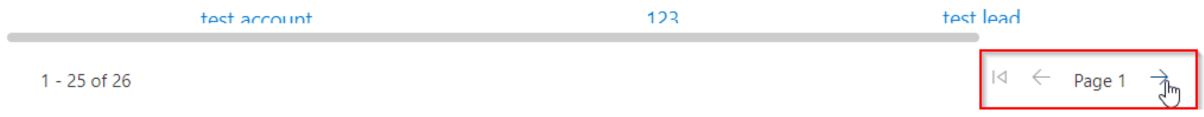
Account Name ↑	Main Pho...	Address 1...	Primary Contact
Lead Test			
Lead Test Account			
New Name	111111		
new value			
test			

Grids, unlike views, are not, really, configurable. They are meant to display the list of records, and they will always do it in the same manner, yet they will offer some capabilities such as:

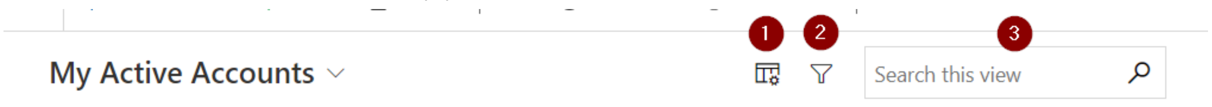
- Ability to sort by clicking on the column title (if you hold "shift" when doing this, you can add columns to the sorting instead of dropping all previously selected one from it)



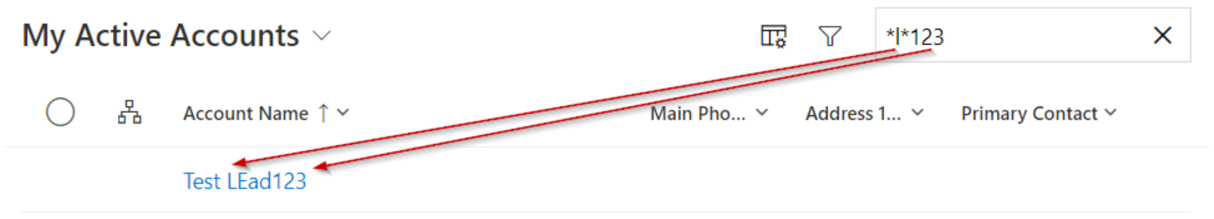
- On the same screenshot above, notice "Filter by" option - there is also the ability to add per-column filters
- Grids will display a certain # of records per page (there is a related configuration settings in the personalization options), and, when there are more pages available, users will be able to navigate to those additional pages:



- Grids also allow adding columns to the view temporarily (since, for example, system views cannot and should not be updated that easily) using "Column Options" button (1) below
- Similarly, it's possible to temporary apply more complex filters filters by using "Advanced Filtering panel" (2)
- And, also, there is a search box (3)



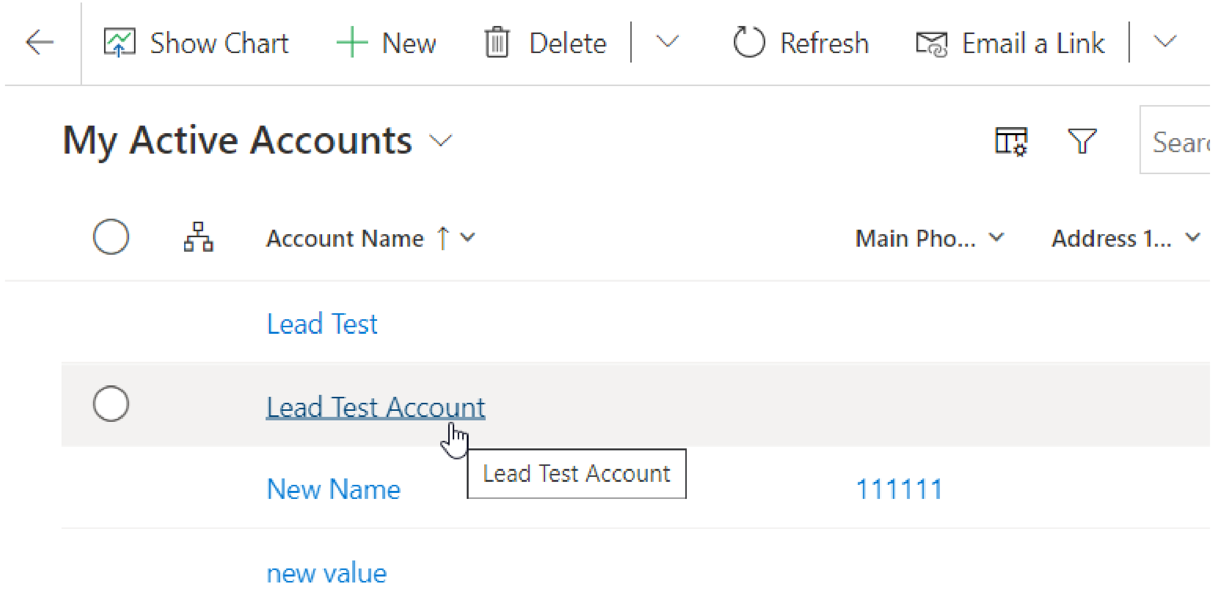
That search box (3) above, also called "quick find", supports wild-card searching as well, so we can use "*" character to specify "any" text. Here is an example:



In order for quick find to look for the data in a specific column, that column needs to be added as a "find column" to the special "Quick Find" view. This view is shared between all applications in the environment, and it has to be configured by the application makers/system administrators. A regular application user does not have control of hose that view is set up.

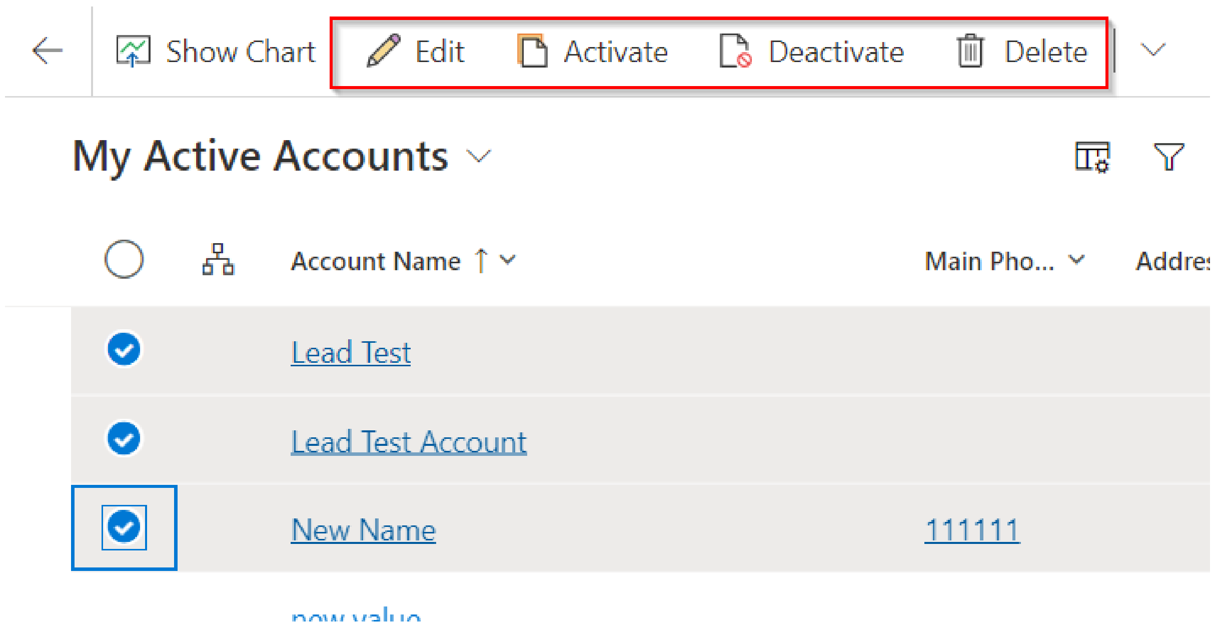
With the above, it's quite possible that, if a column has not been added to the "Quick find" view, a certain value might show up in that column when looking at the unfiltered data, but the grid will produce no search output when searching on that value.

Last but not least, grids will display links for every lookup record, and they will display a link for the "primary" column of the table:



Users can click those links to open such records.
It's also possible to select one or more records (in the latter case, they all have to be on the same page), and, then, apply available contextual command bar actions to the selected records.

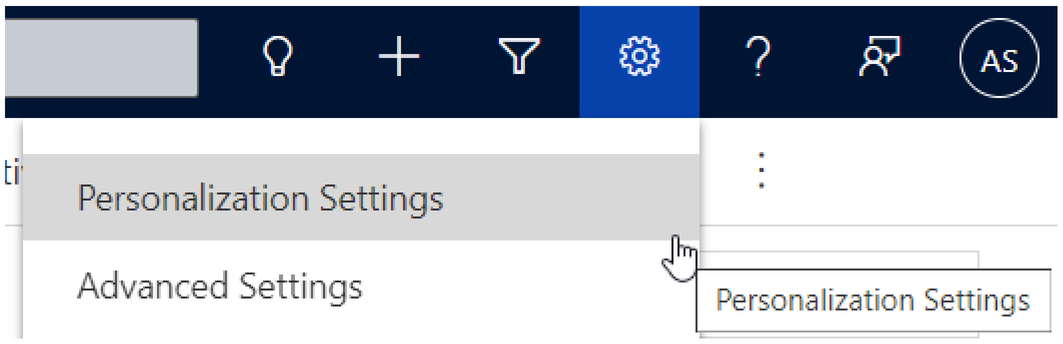
For example, on the screenshot below three different account records have been checked off, and there are “Edit”, “Activate”, “Deactivate”, “Delete” buttons available in the command bar for this selection:



Now, normally we would say “accounts list”, “contacts list”, etc when talking about a list of records. Personally, I find it a good alternative to a somewhat more platform specific “view” / “grid” terminology, especially when that terminology has not been fully adopted by a group, yet.

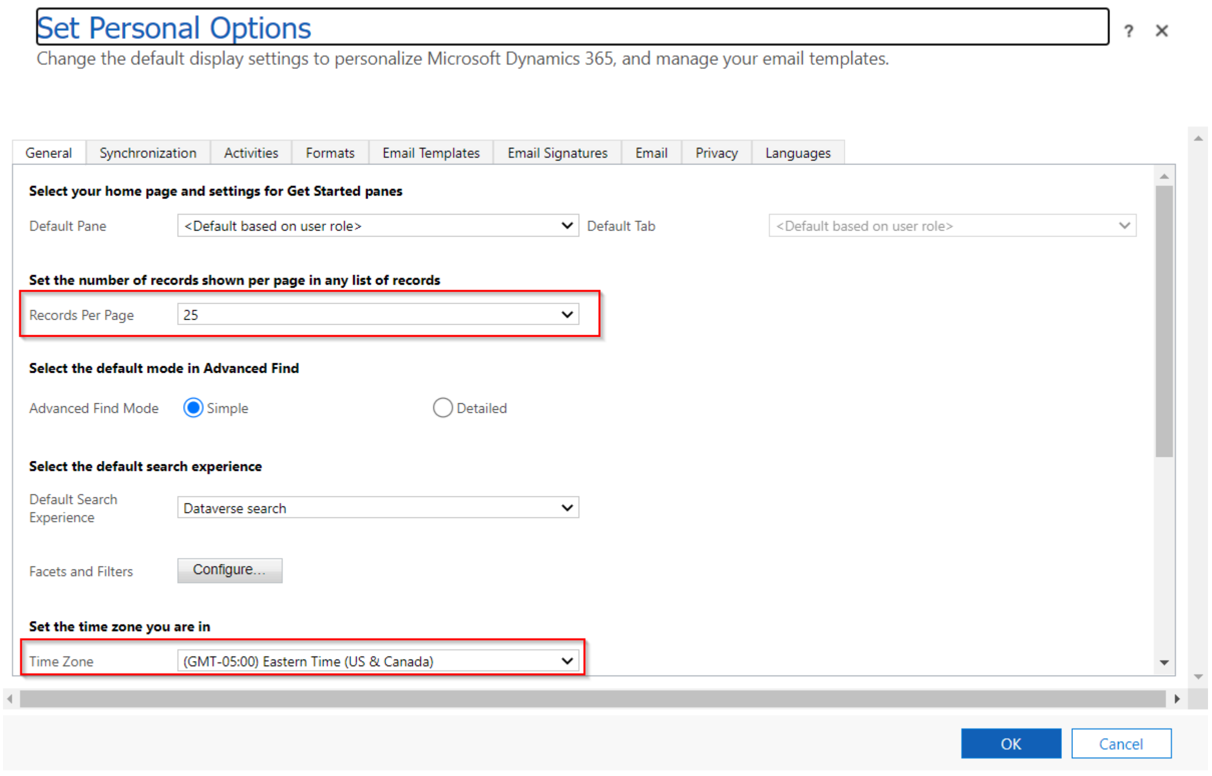
6. Personalization Settings

There are a few personal settings users can adjust in the environments. Those settings are not specific to the application - they are specific to the environments (and remember there could be multiple applications in the same environment).



It is worth looking at all those settings just to be aware of them; however, the first two on the list below, are, probably, the two most widely used:

- Records per page setting affects how many records will be displayed per grid page
- Time zone setting affects the time zone which will be used to convert all time-zone enabled date/time values
- Occasionally, some users might want to change default date, currency, and other formats
- Model-driven applications supports different languages (some translations are available out of the box, others may have to be added as part of setting it all up), so users may decide which of the supported languages they would like to use
- Also, in those environment where email integration has been enabled, users have some control over email synchronization options
- And there are a few other options which can be configured through the personalization settings as well



Set Personal Options

Change the default display settings to personalize Microsoft Dynamics 365, and manage your email templates.

Select your home page and settings for Get Started panes

Default Pane: <Default based on user role>
Default Tab: <Default based on user role>

Set the number of records shown per page in any list of records

Records Per Page: 25

Select the default mode in Advanced Find

Advanced Find Mode: [✓ Simple] Detailed

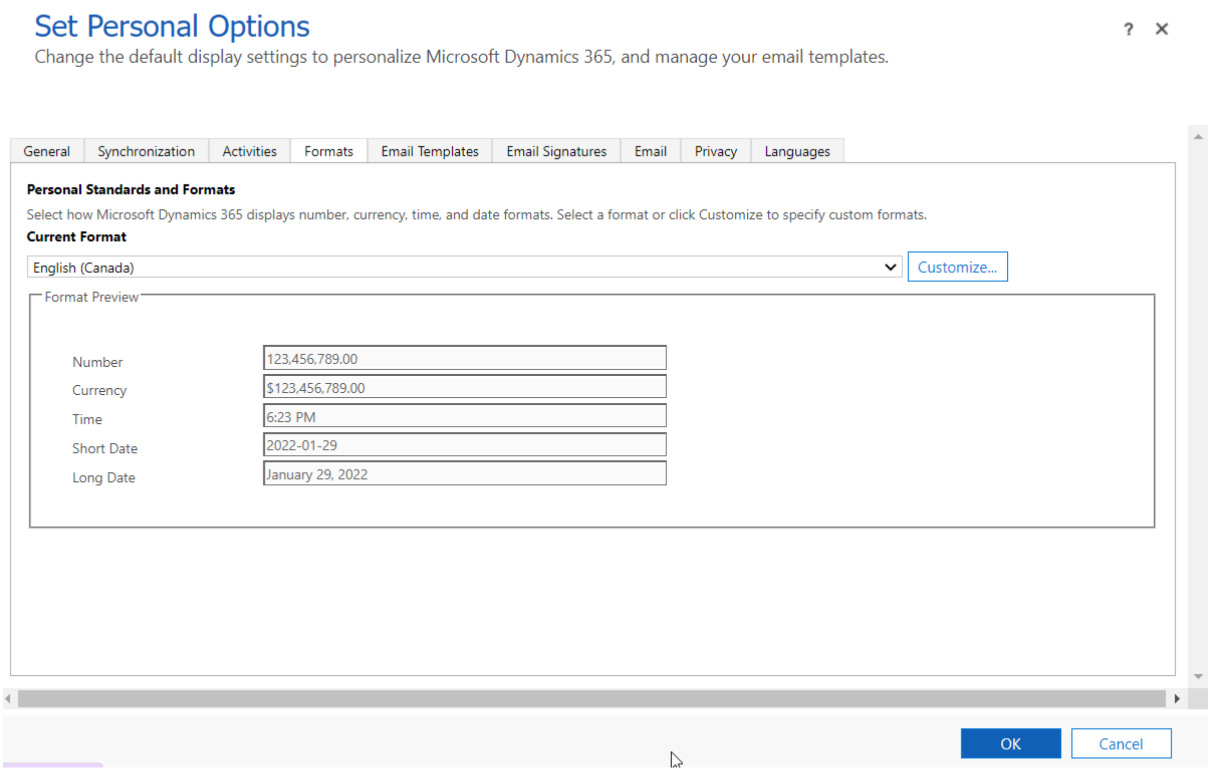
Select the default search experience

Default Search Experience: Dataverse search

Facets and Filters: Configure...

Set the time zone you are in

Time Zone: (GMT-05:00) Eastern Time (US & Canada)


Set Personal Options

Change the default display settings to personalize Microsoft Dynamics 365, and manage your email templates.

Personal Standards and Formats

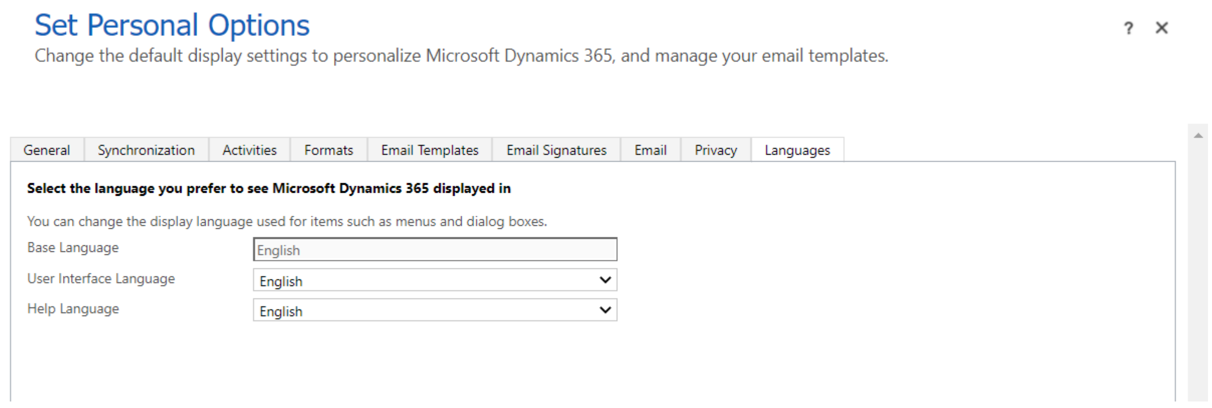
Select how Microsoft Dynamics 365 displays number, currency, time, and date formats. Select a format or click Customize to specify custom formats.

Current Format

English (Canada)

Format Preview

Number	123,456,789.00
Currency	$123,456,789.00
Time	6:23 PM
Short Date	2022-01-29
Long Date	January 29, 2022

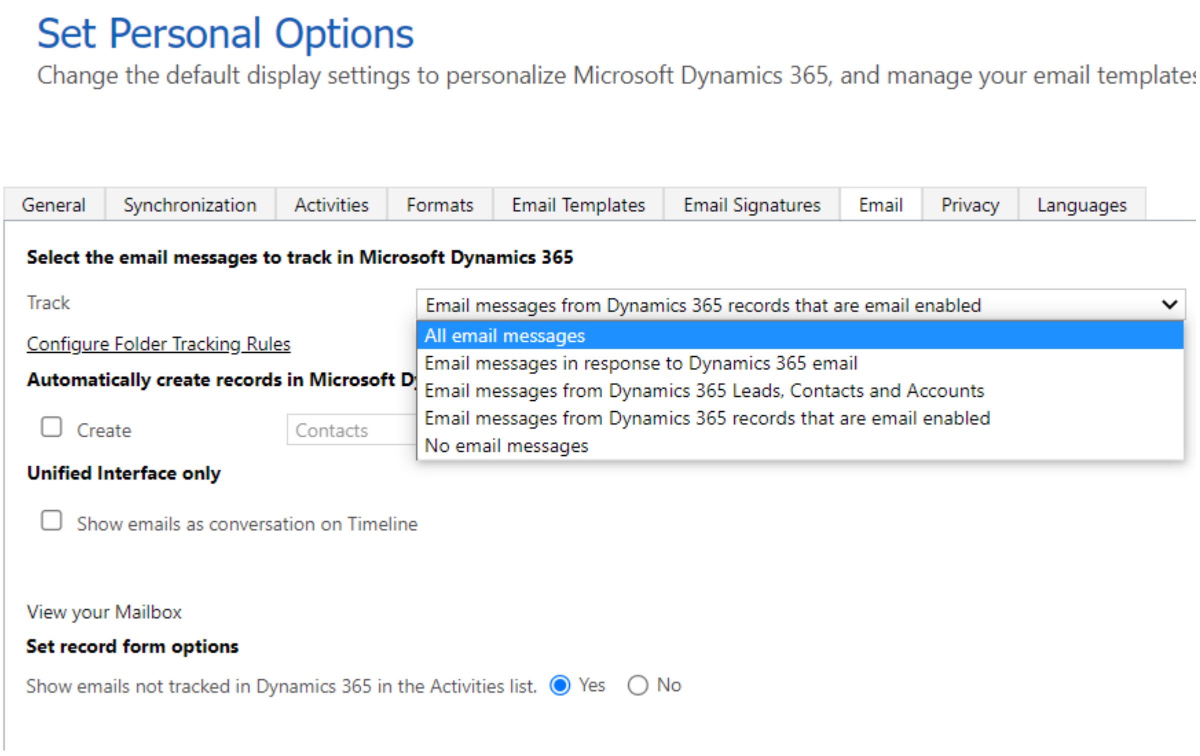

Set Personal Options

Change the default display settings to personalize Microsoft Dynamics 365, and manage your email templates.

Select the language you prefer to see Microsoft Dynamics 365 displayed in

You can change the display language used for items such as menus and dialog boxes.

Base Language	English
User Interface Language	English
Help Language	English



Set Personal Options

Change the default display settings to personalize Microsoft Dynamics 365, and manage your email template

Select the email messages to track in Microsoft Dynamics 365

Track: Email messages from Dynamics 365 records that are email enabled

- All email messages
- Email messages in response to Dynamics 365 email
- Email messages from Dynamics 365 Leads, Contacts and Accounts
- Email messages from Dynamics 365 records that are email enabled
- No email messages

Configure Folder Tracking Rules

Automatically create records in Microsoft D

☐ Create Contacts

Unified Interface only

☐ Show emails as conversation on Timeline

View your Mailbox

Set record form options

Show emails not tracked in Dynamics 365 in the Activities list. [✓ Yes] No

7. Working with the data

Typically, here is how how you will be working with the model-driven applications:

- Open application in the web browser
- Navigate to the desired area using navigation links on the left
- Review list of records displayed in the content area, possibly use various search options to find the one you need
- Use command bar to apply some actions to the existing records, or, possibly, to start creating a new one instead

So far, we have looked at the first 3 steps. In one of the following chapters, we will look at the search and reporting capabilities of model-driven applications a bit more, but, right now, it's the last step above which we need to work through.

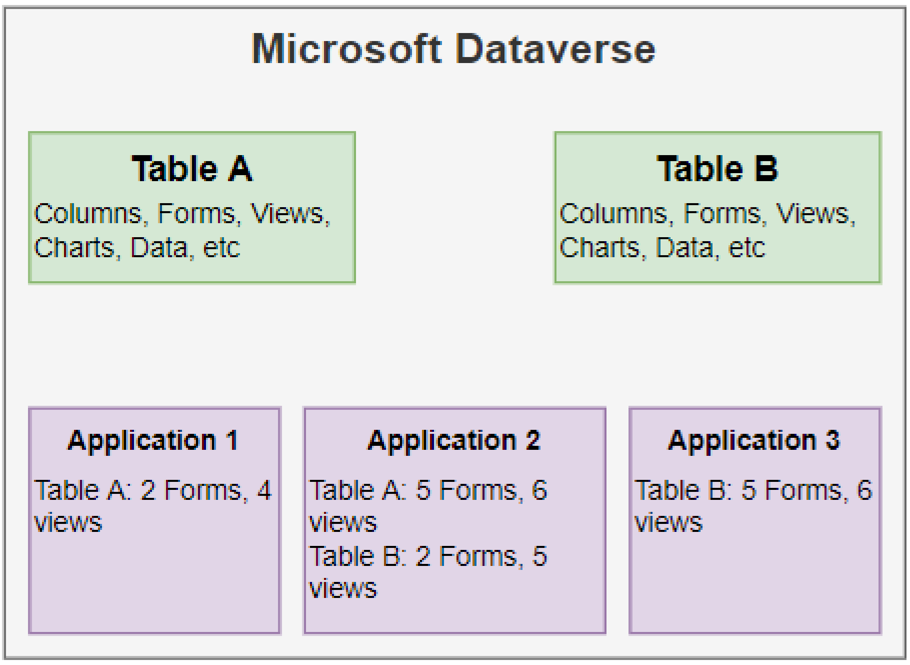
Since, after all, the purpose of model-driven applications is to work with the underlying data, and there is some common functionality offered out of the box. That's exactly what we need to look at in this chapter.

That said, this common functionality can be extended, customized, and it can also be modified by Microsoft every now and then. Still, let's have a look at some of the things to be aware of.

A record in the model-driven application is, essentially, a row in the table. What is a table? I guess you'd be only asking this if you are coming looking at all of this more from the "user" perspective, so "developers" can skip the explanation below.

Anyways, imagine an excel spreadsheet. In the first tab there, you might be maintaining a list of accounts. In the second tab, you might be maintaining a list of contacts. And so on. The simplest analogy would be to say that the whole spreadsheet is a "database", and each tab is a "table". In each of those tables, every row/record would have the same columns, but, of course, rows that belong to different tables would have different columns.

Model-Driven applications can only work with one specific database, which is called Microsoft Dataverse. We talked about tables, records, views… a lot of those and other concepts are attributes of Microsoft Dataverse - in reality, model-driven applications are nothing but containers which combine different components from Microsoft Dataverse:


Microsoft Dataverse
Table A
Columns, Forms, Views, Charts, Data, etc
Table B
Columns, Forms, Views, Charts, Data, etc
Application 1
Table A: 2 Forms, 4 views
Application 2
Table A: 5 Forms, 6 views
Table B: 2 Forms, 5 views
Application 3
Table B: 5 Forms, 6 views

On the screenshot above, you can see how three different applications are using the same tables/forms/views in different combinations. This is not everything that can be included into an application, but the purpose of the diagram above is to illustrate the fact that model-driven applications are simply offering a way to group Dataverse functionality into logical pieces, yet, on top of that, they are offering left-hand navigation which we looked at before.

So, then, when talking about data records in the model-driven applications, keep in mind that we are actually talking about the data stored in Microsoft Dataverse, and, across all the applications in the same environment, different users can perform just the same operations against the same data (as long as that data is exposed in the specific application, and as long as respective users have required security roles)

Here are some of the most typical things you can do with the data records:

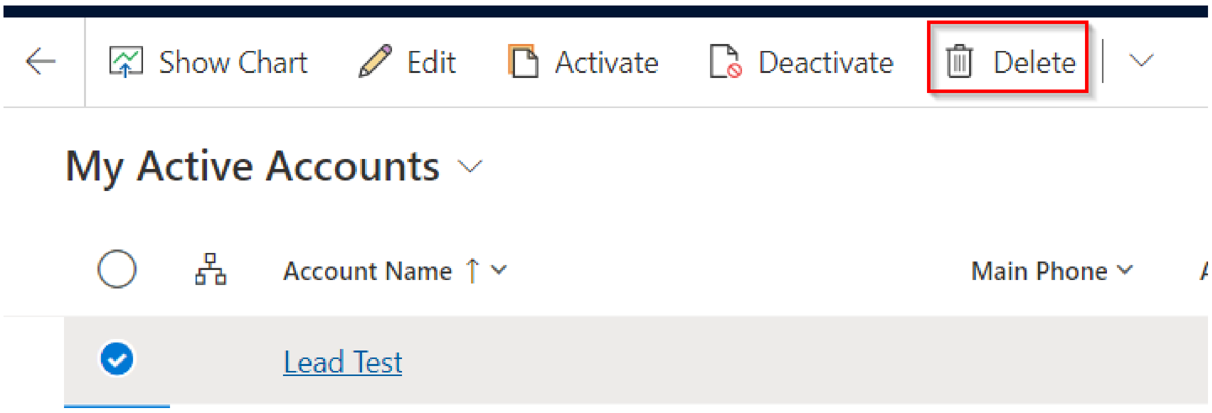
- Perform a search (we looked at it before, we'll look at it again later)
- Activate/Deactivate/Change status
- Assign to a user/team or share with a specific user/team (for some tables, there are no such options)
- Delete a record
- Create a new record

- Update a record
- Associate a record to a record from the same / other table

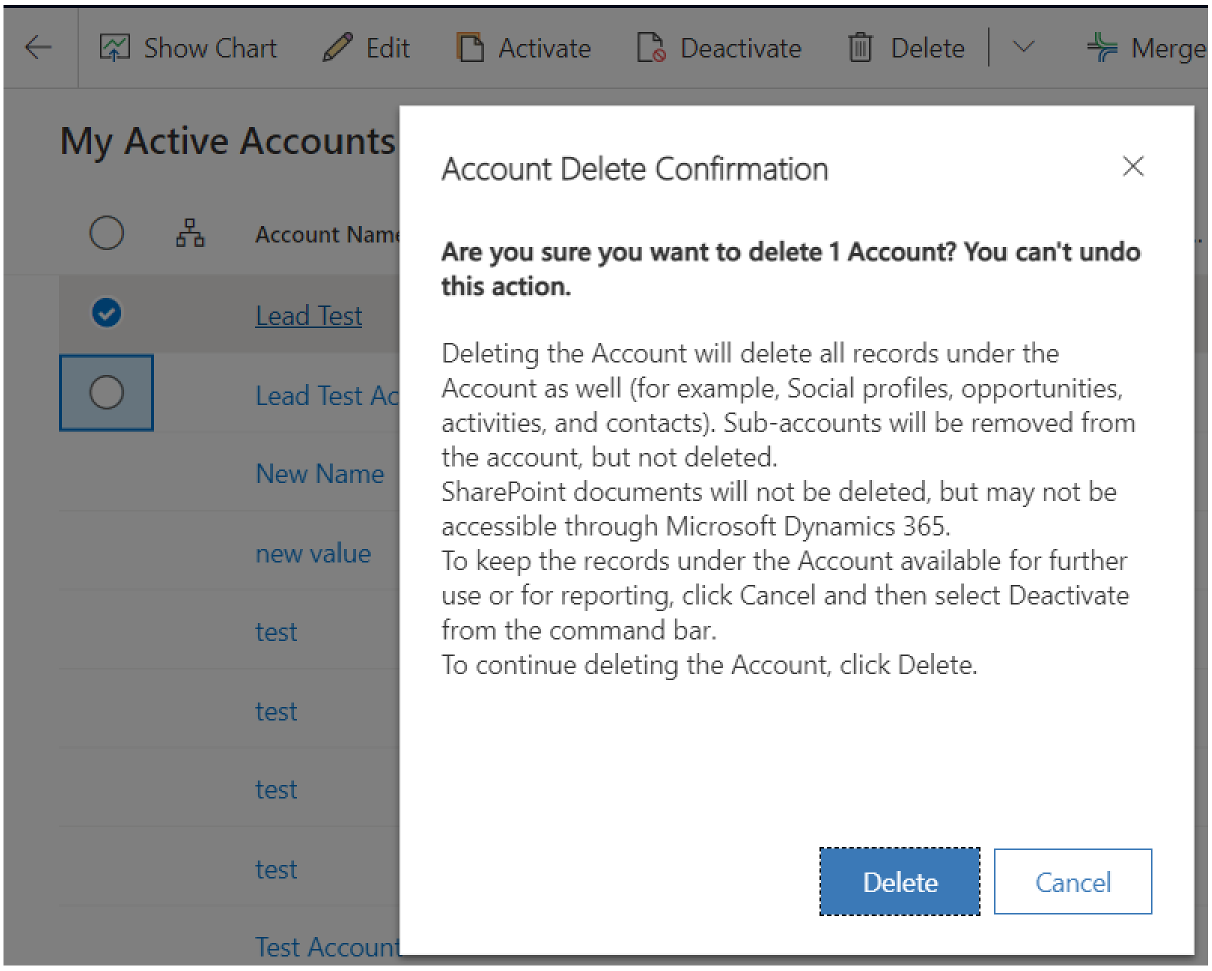
User’s ability to perform those operations is controlled through the security roles assigned to the user. Apparently, system administrators will be able to do all of that. However, it’s not unusual that “Delete” operation will be disabled through the security roles, and, so, regular users might not be able to delete any records.

Why would application makers decide to limit users’ ability to delete data? Well, there are lots of considerations there, but data integrity and history tracking is, likely, the most important ones. If everybody could delete data freely, intentionally or not, it would make it almost impossible to see the history in the context of some business processes. Besides, that data may be shared - for example, imagine a contact which is a “customer” on two different opportunity records is deleted by mistake. There would be two opportunity records in the system, and none of them would have contact details to reach out to the “client”. You may think this won’t happen, but, in reality, it is almost bound to happen. Sooner or later, somebody will delete a record which is not supposed to be deleted, and that will mean trouble.

This is why, when looking at the command bars, you might not be able to see “delete” button:



But, if you do see it, you might try using it. In case with the account records, for example, the system will give you an enhanced warning message:

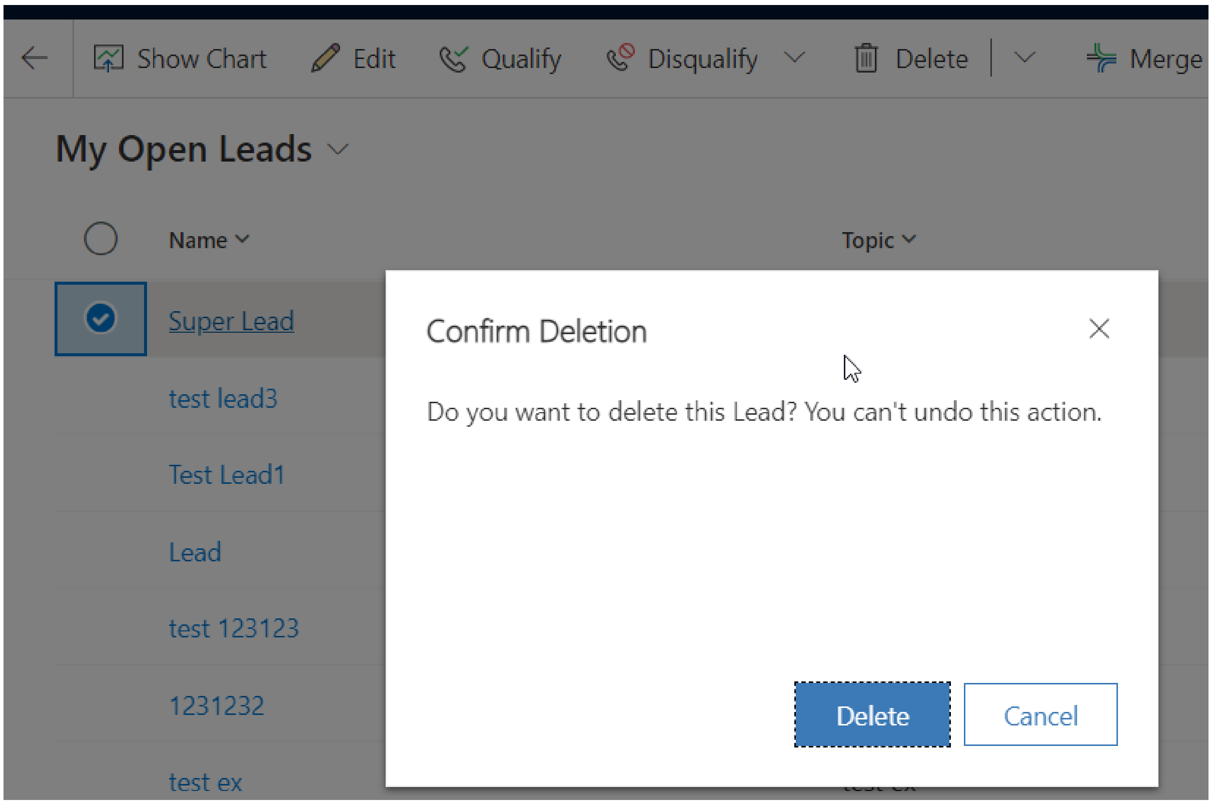


Account Delete Confirmation

Are you sure you want to delete 1 Account? You can't undo this action.

Deleting the Account will delete all records under the Account as well (for example, Social profiles, opportunities, activities, and contacts). Sub-accounts will be removed from the account, but not deleted.
SharePoint documents will not be deleted, but may not be accessible through Microsoft Dynamics 365.
To keep the records under the Account available for further use or for reporting, click Cancel and then select Deactivate from the command bar.
To continue deleting the Account, click Delete.

But, if you try it with most of the other record, you'll see a much more simple confirmation dialog:



Confirm Deletion

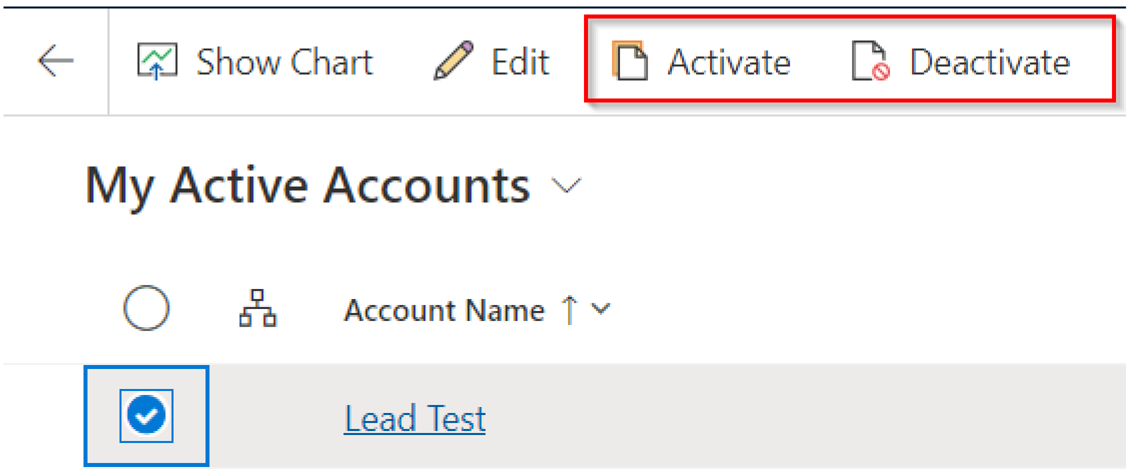
Do you want to delete this Lead? You can't undo this action.

Having talked about the “Delete” operation, let’s talk about what is called “soft delete” sometimes.

Instead of deleting a record, it is, often, better to just deactivate it. Inactive records become readonly in the model-driven applications (behind the scene, they can still be updated), yet they would normally disappear from the views, since most of the views would have a condition to filter out “inactive” records.

Essentially, inactive records will almost disappear, but they will still be there in case they are needed and/or for historical purposes.

If a record that has been deactivated previously needs to be brought “back to live”, it can be activated. All of this is, usually, done using “Activate” / “Deactivate” buttons:

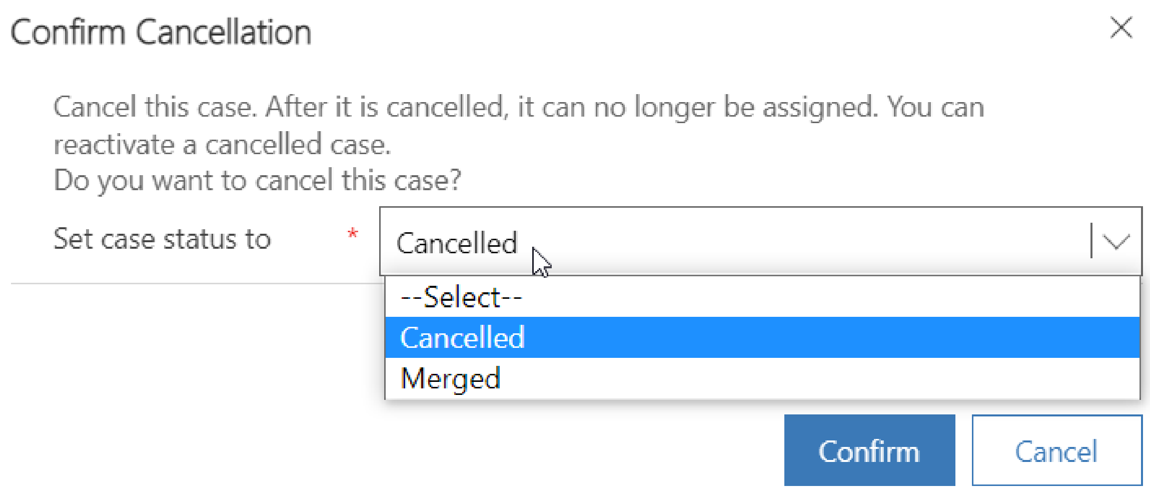


Sometimes, those buttons may be called differently (for example, lead records are not "deactivated", they are "disqualified").

Sometimes, those buttons might be hidden by default (for instance, there is no "Activate" button for tasks - it's hidden out of the box, but it can be re-added by the application makers / developers).

In fact, "Activate" and "Deactivate" buttons are just two examples of how statuses and status reasons work in Microsoft Dataverse (and, therefore, in the model-driven applications).

Every now and then, you will click "Deactivate", and you will see a popup dialog suggesting to choose from one of the "reasons". For example, when cancelling a case ("Cancel" button, which is an equivalent to "Deactivate"), you see this kind of popup window:


Confirm Cancellation

Cancel this case. After it is cancelled, it can no longer be assigned. You can reactivate a cancelled case.
Do you want to cancel this case?

Set case status to * Cancelled

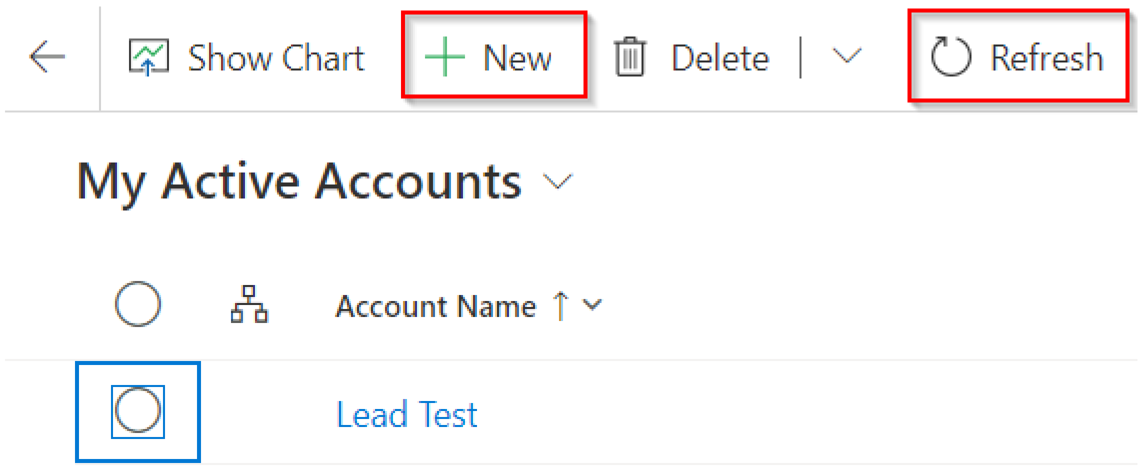
Every record in Microsoft Dataverse will have “Status” and “Status Reason” columns. They are related in the sense that for every “Status” there can be one or more “Status Reasons”. Those are defined per table, they can be different for each table, and model-driven applications are offering functionality to work with those out of the box.

Aside from “Activate”, “Deactivate”, and “Delete”, there are, usually, other buttons in the command bars.

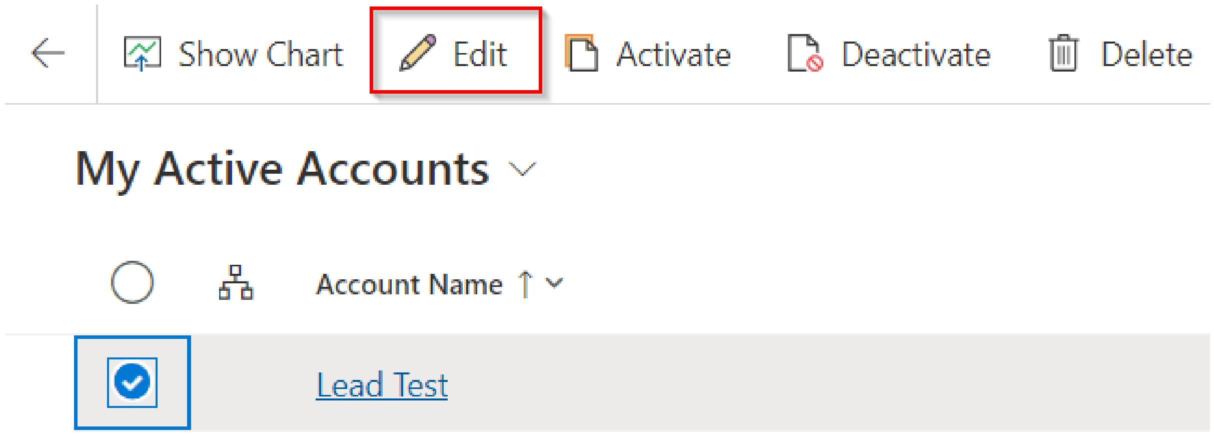
There are three more I wanted to touch quickly:

Actually, there are two on the screenshot above. “Refresh” does exactly what it is meant to do - it is refreshing the content area, be it a grid, or be it an individual form (which we will look at in the next chapter).

“New” is there so users could start creating a new record.



However, since command bards display buttons in the context of what's happening in the "content" area, "Edit" will be displayed there once a record has been selected in the grid, as you can see on the screenshot below:

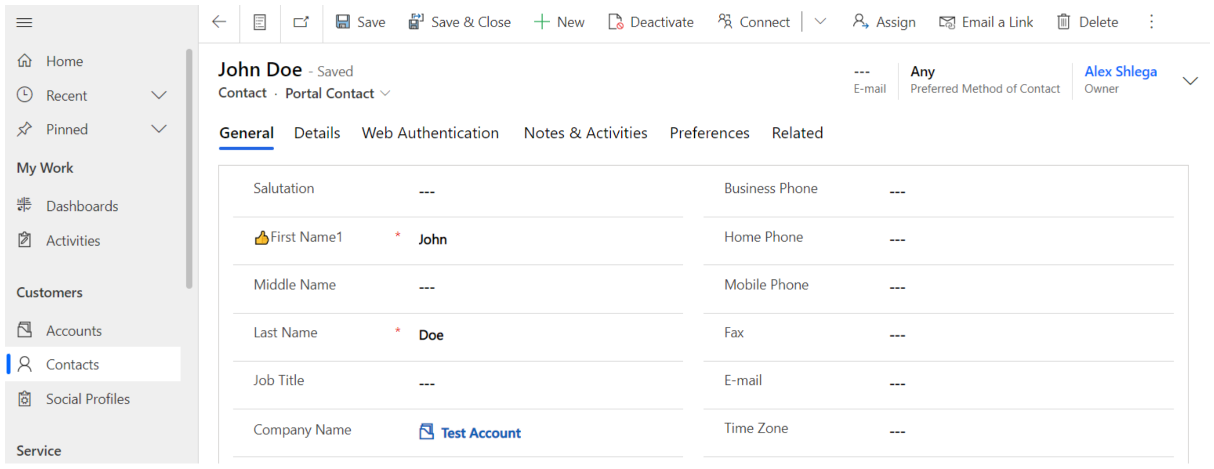


In the next chapter, we'll talk about the "forms", which is where you can update individual columns for a given record if you opened it in the application using "Edit" button, or where you can provide values for the new record on essentially the same form if you opened it using "New" button instead.

8. Forms

Forms in the model-driven applications are simply screens where users can look at the details of the individual records and possibly update some of those details. Forms are also used to create new records.

For example, here is an example of a contact form - notice there is, also, a command bar on that screen:



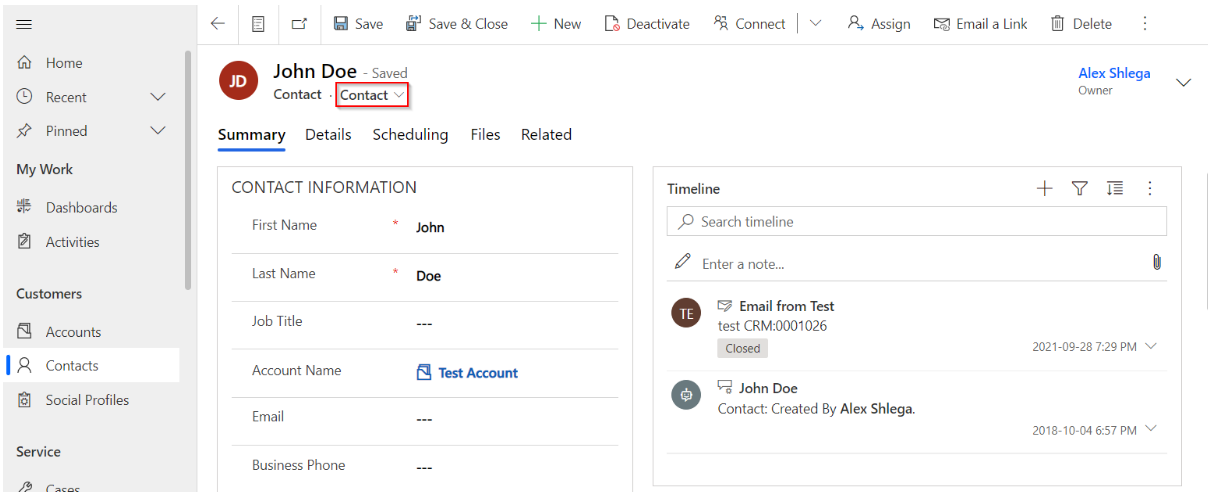
John Doe - Saved
Contact · Portal Contact

E-mail	---	Preferred Method of Contact	Any
Owner	Alex Shlega		

Salutation	---	Business Phone	---
First Name1 *	John	Home Phone	---
Middle Name	---	Mobile Phone	---
Last Name *	Doe	Fax	---
Job Title	---	E-mail	---
Company Name	Test Account	Time Zone	---

Application makers / developers can create and update table forms so users can, then, utilize those form when working with the application. It is worth mentioning that forms can be re-used across multiple applications in the same environment, but, at the same time, some applications may choose to use specific forms only (the same is true for the views, by the way).

For example, here is a different form which is displaying the same contact record:



JD John Doe - Saved
Contact · Contact

Alex Shlega
Owner

CONTACT INFORMATION

First Name *	John
Last Name *	Doe
Job Title	---
Account Name	Test Account
Email	---
Business Phone	---

TE Email from Test
test CRM:0001026
Closed
2021-09-28 7:29 PM

John Doe
Contact: Created By Alex Shlega.
2018-10-04 6:57 PM

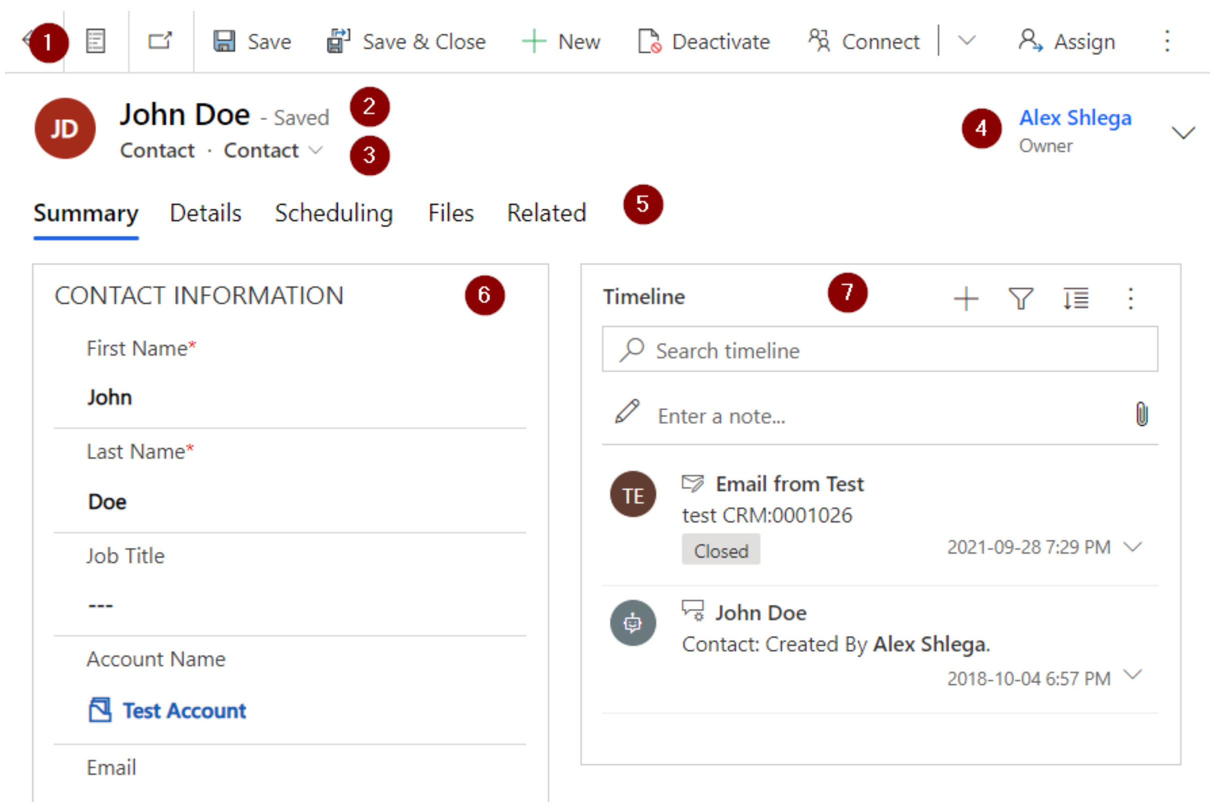
If a user working with the application has access to multiple forms through the security roles, and if more than one of those forms has been added to the application by the developers, there will be a form selector dropdown (highlighted above) to choose from one of the available forms.

This might, actually, lead to the occasional confusion, since Model-Driven applications do remember the latest form selection per user (but not per record), so the next time a record from the same table is opened in a form, model-driven application will use the same form that the user selected before.

Every table may have hundreds of columns, lots of relationships, etc. Not all of those are needed when creating a new contact through the call centre. On the other hand, when creating or updating a portal contact, portal administrators may need to see portal authentication methods for that contact.

And those are just two examples, but, in general, application makers will use multiple forms to optimize data entry for different scenarios.

Let’s look at some of the common elements you will see on the forms:



First Name*

John

Last Name*

Doe

Job Title

Account Name

Test Account

Email

TE Email from Test
test CRM:0001026
Closed
2021-09-28 7:29 PM

John Doe
Contact: Created By **Alex Shlega**.
2018-10-04 6:57 PM

There are some special types of forms, but, most of the times, here is what you'll see:

- There will be command bar (1)
- There will be record name (2)
- There will be form selector (3) if more than one form is available
- There might be header (4) with a few columns displayed there
- There will be "tab" navigation (5)
- There will be 1 or more sections (6), (7) with columns and subgrids

Command bar should look familiar now; although, there will be some buttons which are not available in the "grid" command bards.

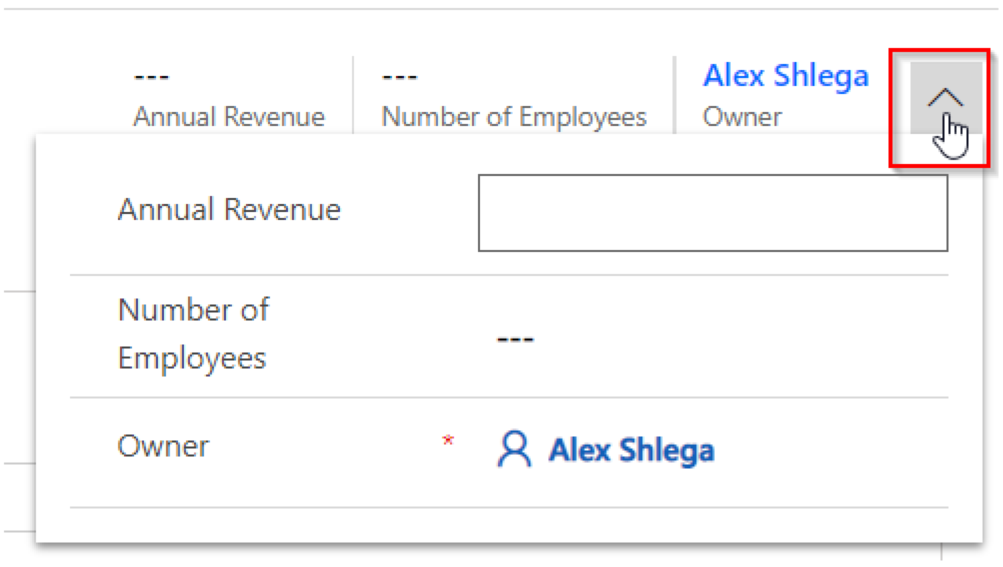
Specifically, there will be "Save" and "Save and Close" buttons among the other ones. Those two will show up only when the record being displayed in the form is editable. "Save" will save the changes. "Save and Close" will also close the form after that, and the application will navigate back to the previous screen (it could be a different form, or, possibly, a view, etc). "Save and Close" is very handy when a user does not need to do anything else with the current record, and, instead, just needs to save it and go back to where they came from in the application.

Record name (2) is there only for “display” purposes - it’s not editable. Depending on the table, it could correspond to different columns (Subject, Title, Name, etc), and, in order to give application users ability to update record names, that column would have to be added to one of the sections on the form.

We have already looked at the form selector (3) - again, that’s there to give access to multiple forms. Some applications will expose only a single form, in which case there will be no form selector while in that application. Also, some users will have access to only one form, which will lead to the same result.

Header, Tabs, and Sections are there to help organize columns on the screen.

There is only one header per form, and it’s displayed at the top right corner. Headers may display a few columns, so, in that sense, they are similar to form sections. However, unlike with the sections, in order to start entering column values into the input controls in the header, header will need to be expanded:

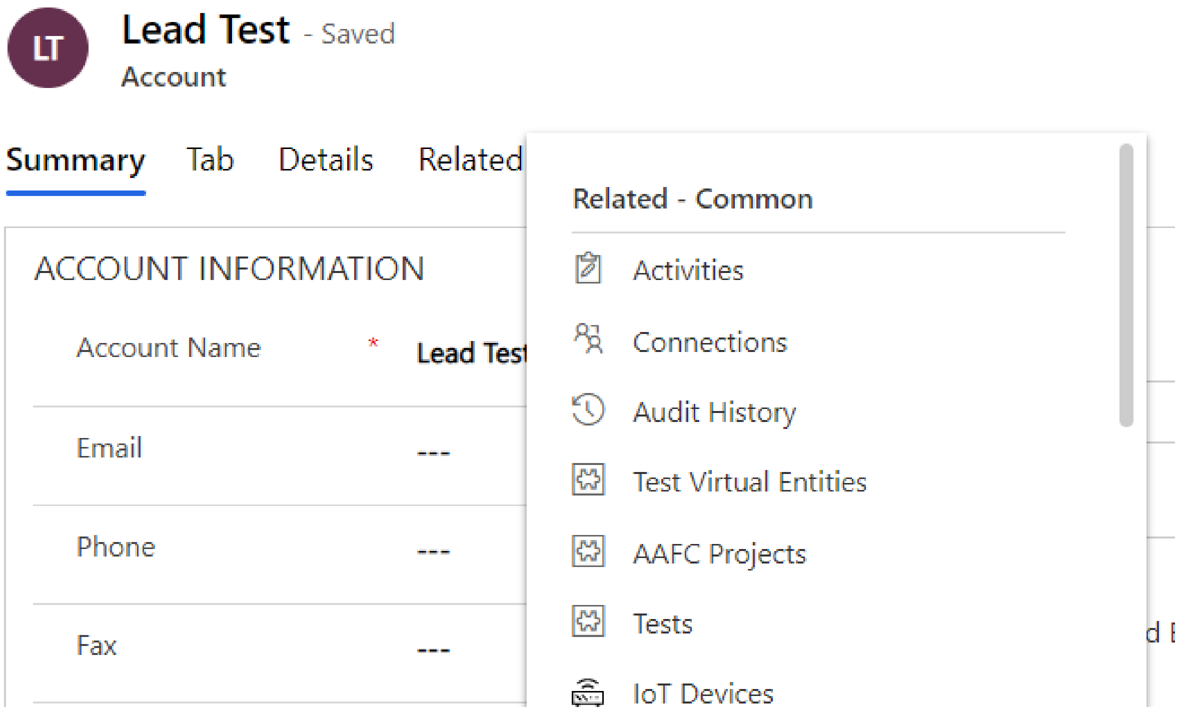


As for the tabs, they group different sections together, and only one tab is visible on the screen at any time. There is, also, tab navigation at the top of the form. So, for example, application makers could choose not to display hundreds of columns on the same tab/section, since it might be a long screen then, and, instead, organize them into multiple sections, then organize sections into tabs, thus making the whole form screen easier to navigate through.

Note: form design and configuration is for the application makers to do. Regular users don't have access to the form designer functionality.

There could be multiple tabs, and there are no prescribed names for them, but there is only special item in the tabs navigation area called “Related”. That's not, really, a tab, but it behaves a bit as a tab.

What it does is once clicked is: it shows all related tables which are linked to the current table through a relationship:



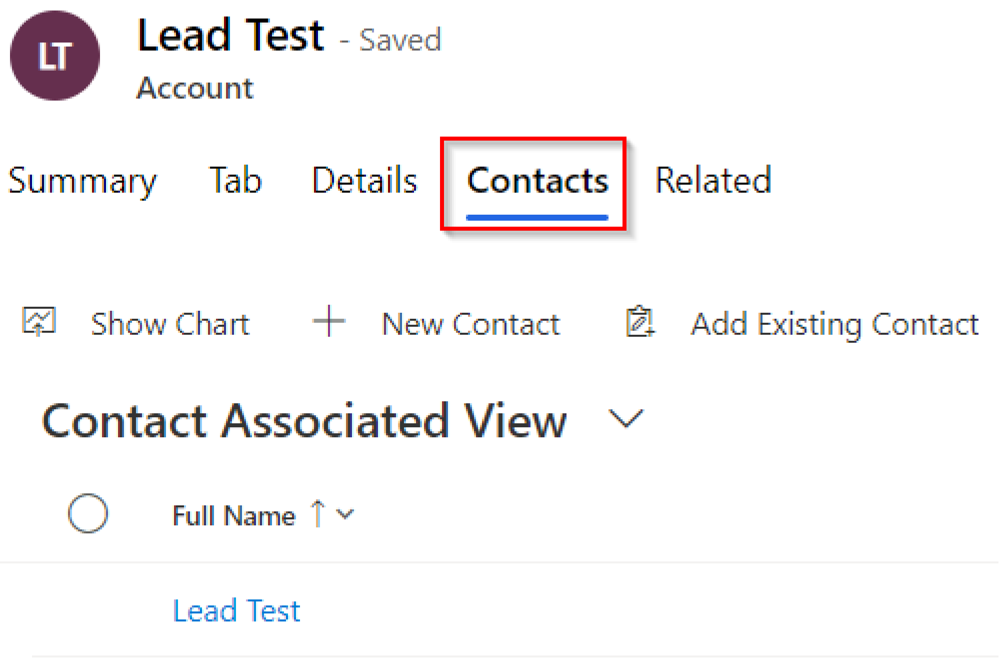
There has to be a relationship in the Microsoft Dataverse, and it has to be a 1:N or an N:N relationship.

Note: what is a relationship? It is a way of specifying that two tables are related to each other in some way. 1:N means Parent-Child relationship, where, for example, you may have an organization record in the system, and you may have different contact records all related to the same organization since it's their "primary account". Although, it could also be N:N if the relationship is not so much about one table being a "parent" and another being a "child" - what if the same contact can be related to different organizations, since that contact represent an individual who is providing consulting services to all those organizations? In which case the same organization may be related to different contacts through the same relationship, of course. This is called N:N.

There is, also, N:1, which is a reversal of the 1:N (so, where there is 1:N, there is, also, N:1)

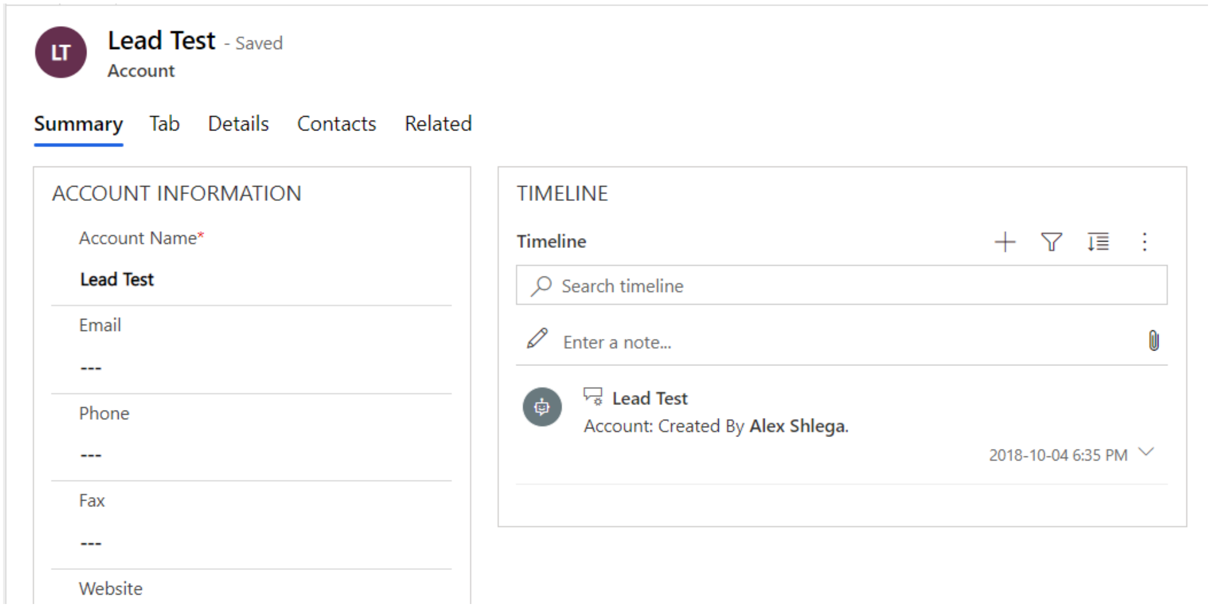
However, on the form screen only 1:N and N:N relationships will be showing up in the "Related" dropdown.

Once a specific related table is selected from that dropdown, a new tab gets added to the form, and related records from the selected table are displayed in that tab:



Keep in mind that those "dynamically added" tabs do not get re-added to the form automatically after the form is closed, and, then, opened again.

With that, let’s just have a few more example of the tabs here - the screenshots below are for the same record and for the same form, but selected tab is different on each screenshot:



ACCOUNT INFORMATION

Account Name*	Lead Test
Email	---
Phone	---
Fax	---
Website	

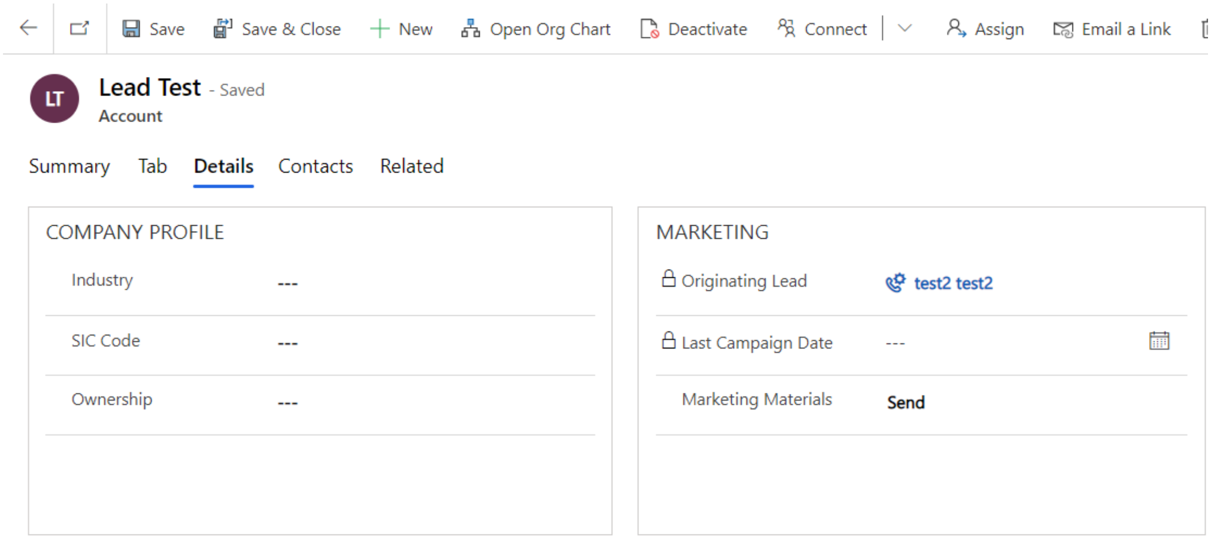
TIMELINE

Timeline

Search timeline

Enter a note...

Lead Test
Account: Created By Alex Shlega.
2018-10-04 6:35 PM



COMPANY PROFILE

Industry	---
SIC Code	---
Ownership	---

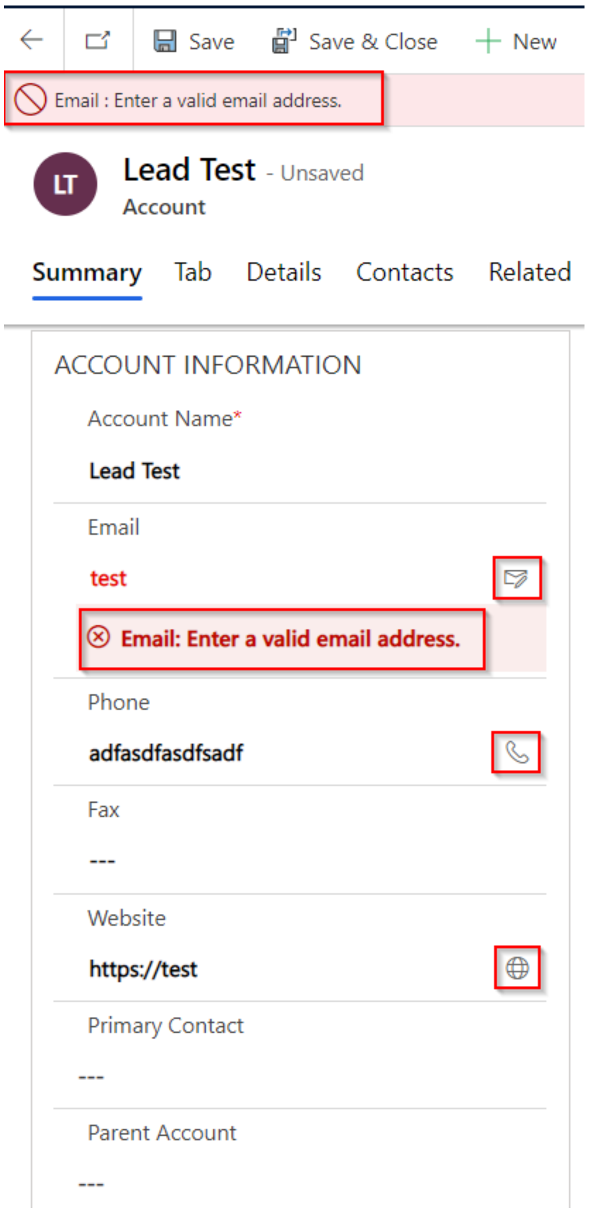
MARKETING

Originating Lead	test2 test2
Last Campaign Date	---
Marketing Materials	Send

If you take another look at the screenshots above, you’ll notice that various columns on the form are grouped into what is called “sections”. Application makers can only put columns inside form sections in the model-driven apps (with the exception of “headers”, but those can be thought of as being just another somewhat special section).

Working with form columns from the user perspective is straightforward - it's all about filling in required data and saving those changes.

Some columns will expect the data to be using specific format, and, when it's not the case, there will be a validation error displayed on the form:



Email : Enter a valid email address.

Lead Test - Unsaved
Account

ACCOUNT INFORMATION

Account Name*
Lead Test

Email
test
Email: Enter a valid email address.

Phone
adfasdfasdfsadf

Fax

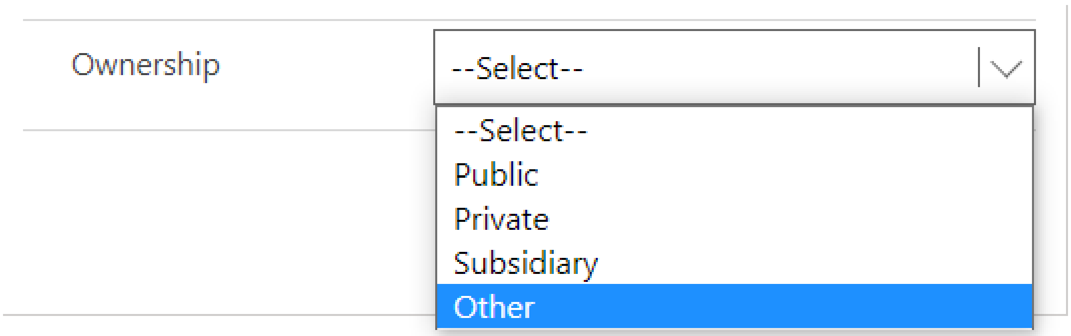
Website
https://test

Primary Contact

Parent Account

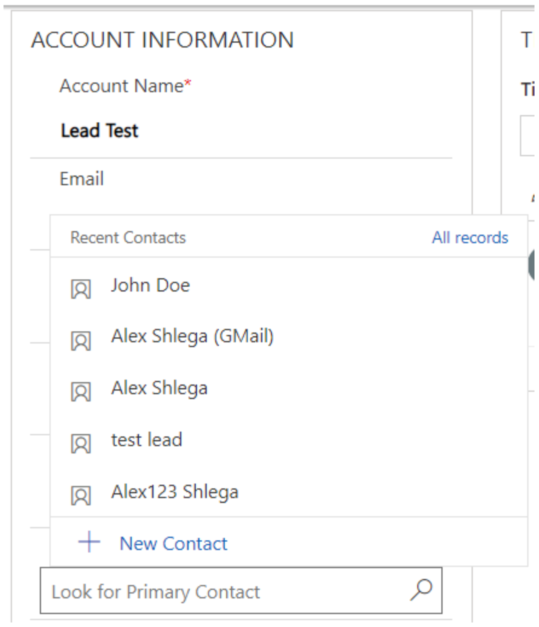
Also, a few special column types support quick actions (for example: sending an email, creating a phone call, opening a web page).

There are “choice” columns:



Those choices are configurable, but any configuration changes to the list of possible values would require certain permissions (System Customizer / System Administrator).

As an alternative to the “choice” columns, there will often be a “lookup” column:



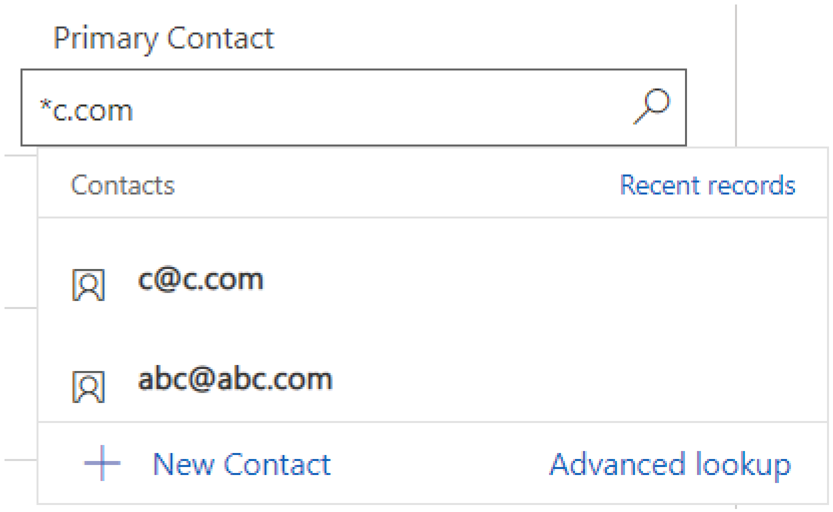
Unlike with the choices, lookup values can be maintained by the regular users. In reality, lookup values are just records from a related table (it could also be a combination of tables), so users’ access to those tables is controlled through the security roles.

From the user interface and usability perspective, choices and lookups are not, always, interchangeable.

Lookup input controls on the form are, actually, relatively complex compared to the choices.

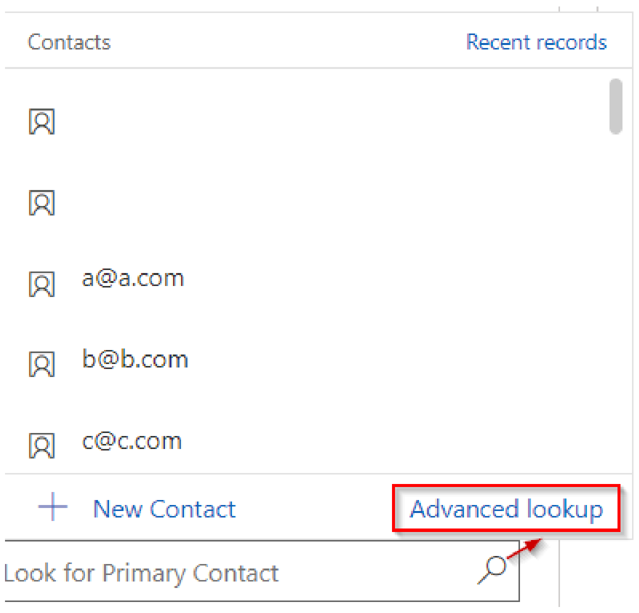
First of all, application makers have control over which view will be used to display possible “lookup values” (as in, “active contacts”, “all contacts”, etc)

Lookup control support wildcard search functionality:

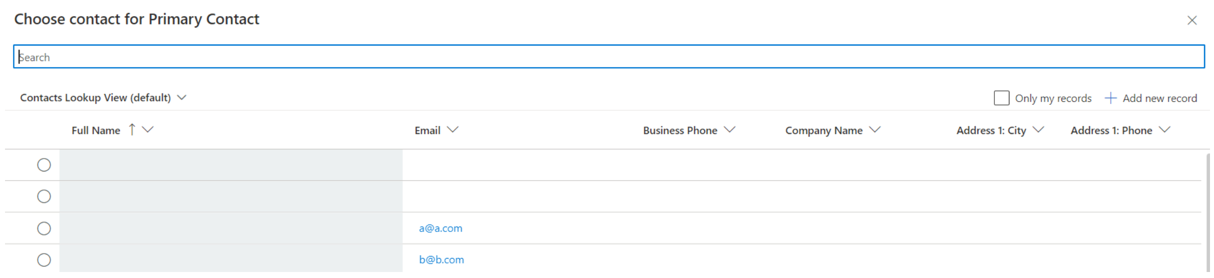


They can show you all records or “recent” records (those you have recently selected).

And, also, there is “Advanced lookup” link:



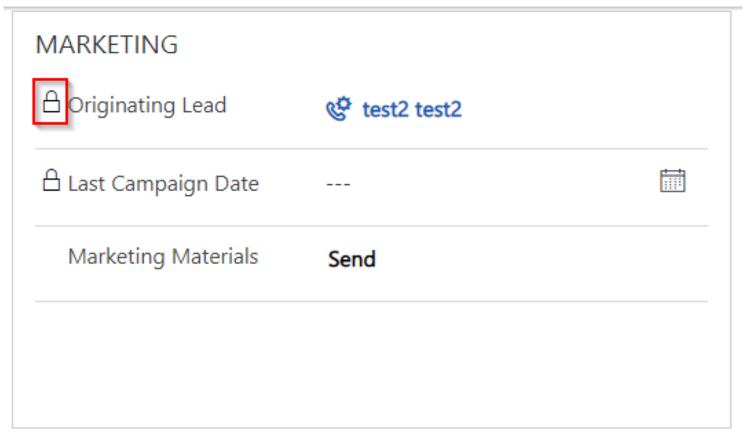
Advanced lookup brings up a more advanced version of the search screen where users can pick a view and see all columns from that view:



Choose contact for Primary Contact
Search
Contacts Lookup View (default)
Only my records
Add new record

Full Name	Email	Business Phone	Company Name	Address 1: City	Address 1: Phone
	a@a.com				
	b@b.com				

No matter what the column represents (date, text, lookup, etc), it can be marked as “readonly” on the form:

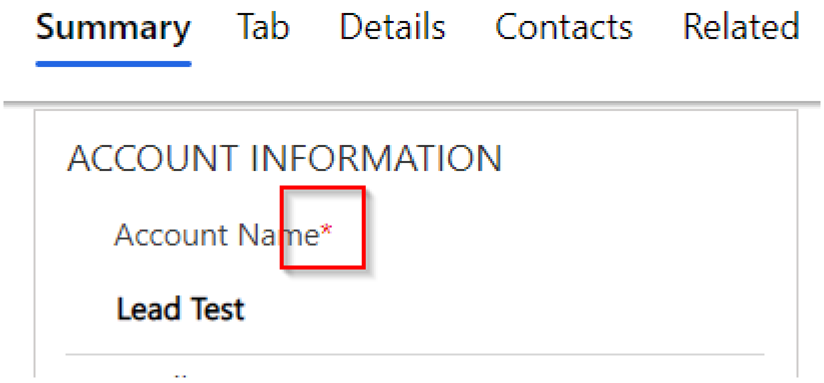


MARKETING

Originating Lead	test2 test2
Last Campaign Date	---
Marketing Materials	Send

Keep in mind that the same column can be read-only in some cases, and it can be editable in others. It can even be read-only on one form and editable on another.

Also, some columns may have a red asterisk next to the label - those are required columns, and, so, they cannot be left empty when creating / updating a record:



ACCOUNT INFORMATION

Account Name*

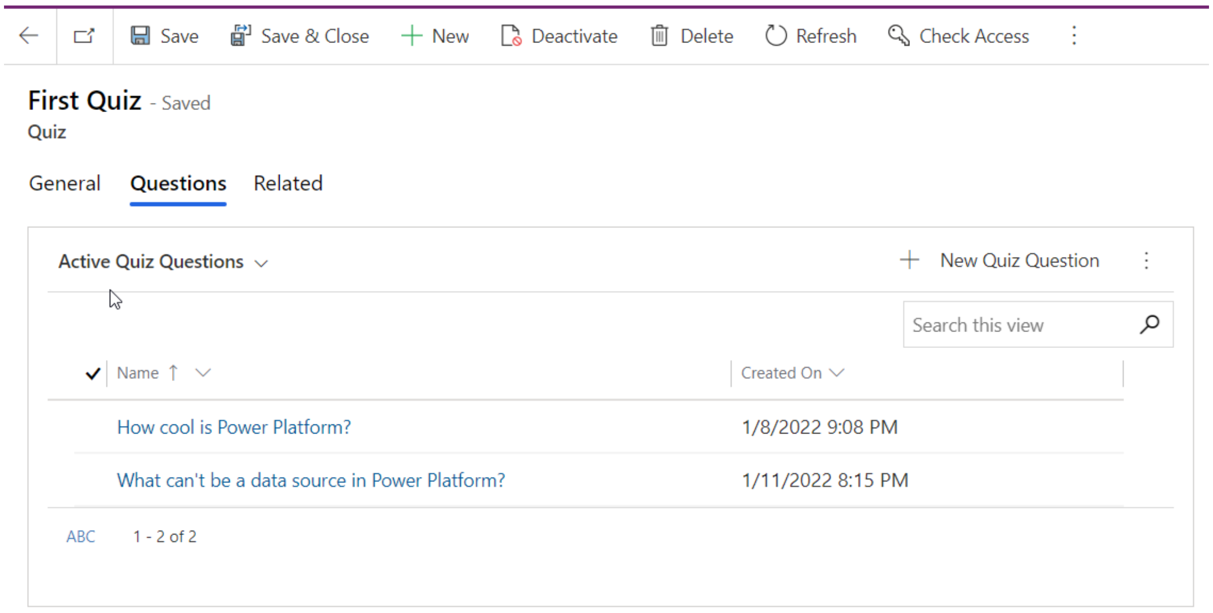
Lead Test

Again, the same column can be required on one form, and it can be optional on another.

Note: there are customizations application makers can apply to the forms to lock and unlock certain columns and/or to make them read-only using conditional logic.

All of the above works more or less the same way while creating a new record or while updating an existing record.

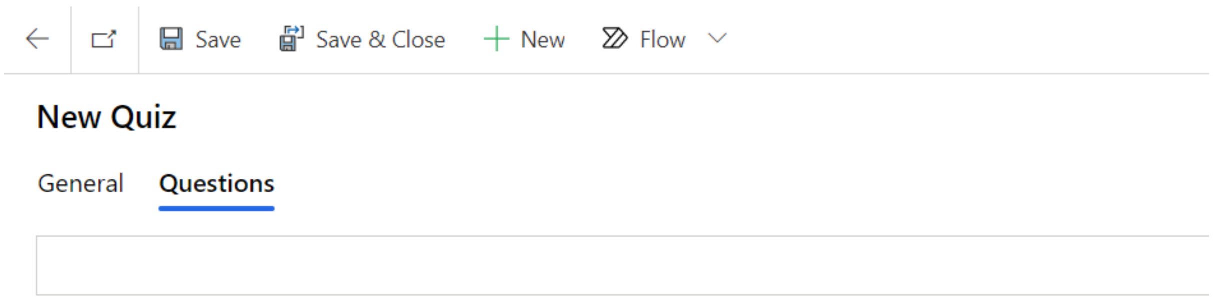
It is a little different for the subgrids, however. Subgrids are just grids displayed on the forms. For example, on the screenshot below there is a “Quiz” records and there are associated “Quiz Questions” which are displayed in a subgrid:



Name	Created On
How cool is Power Platform?	1/8/2022 9:08 PM
What can't be a data source in Power Platform?	1/11/2022 8:15 PM

1 - 2 of 2

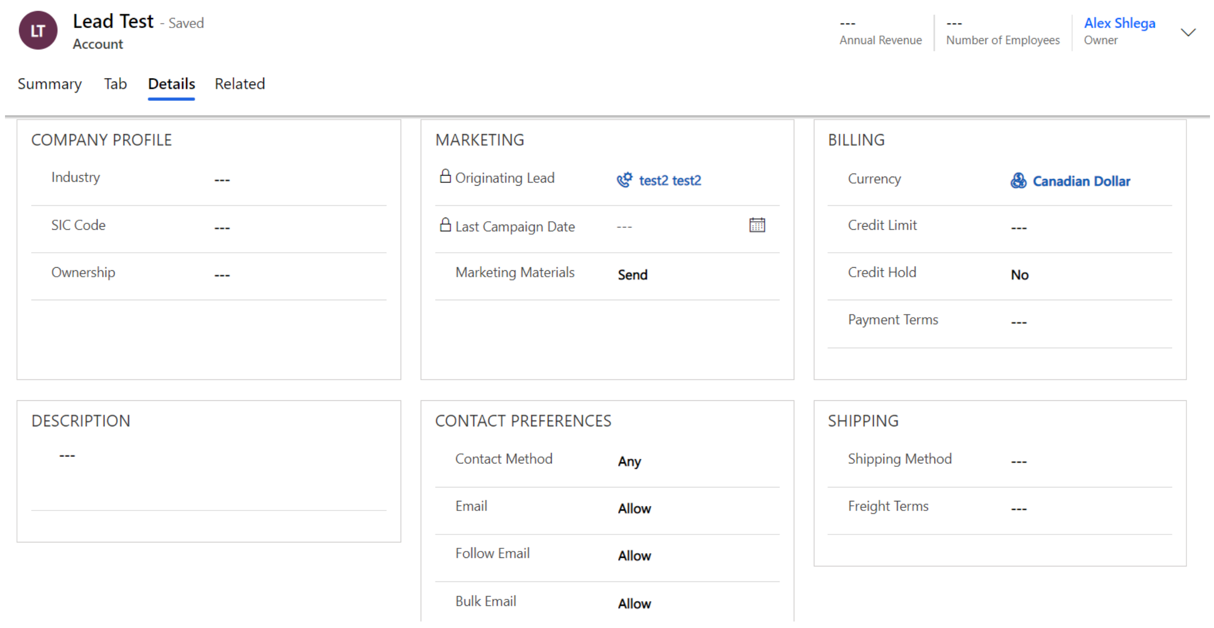
If it were a “new quiz” record, the subgrid would look like this:



The reason for that is that for new records there is nothing to associate related records to, yet, nothing can be displayed in the subgrid. Even more, the subgrids itself cannot be displayed either.

You have probably noticed by this time that the same form can arrange screen elements a little differently depending on the browser window size. This is intentional, since model-driven applications implement responsive design, which means they make an effort to look well on different screens.

For example, here is how account form may look like on a bigger screen:



LT Lead Test - Saved
Account

--- Annual Revenue | --- Number of Employees | Alex Shlega Owner

COMPANY PROFILE

Industry	---
SIC Code	---
Ownership	---

MARKETING

Originating Lead	test2 test2
Last Campaign Date	---
Marketing Materials	Send

BILLING

Currency	Canadian Dollar
Credit Limit	---
Credit Hold	No
Payment Terms	---

DESCRIPTION

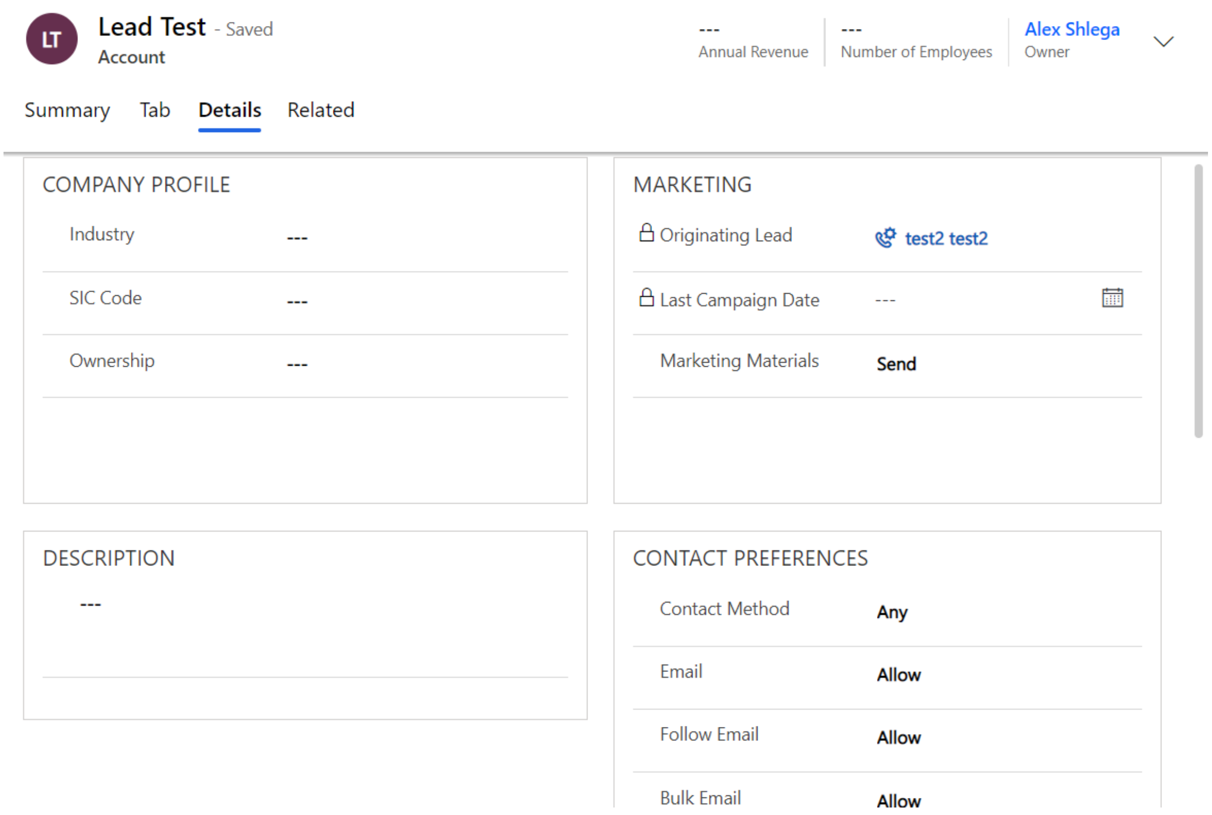
CONTACT PREFERENCES

Contact Method	Any
Email	Allow
Follow Email	Allow
Bulk Email	Allow

SHIPPING

Shipping Method	---
Freight Terms	---

And, when the windows size is smaller, it re-flows in the following way:


Lead Test - Saved
Account

Annual Revenue	---
Number of Employees	---
Owner	Alex Shlega

Summary Tab **Details** Related

COMPANY PROFILE

Industry	---
SIC Code	---
Ownership	---

MARKETING

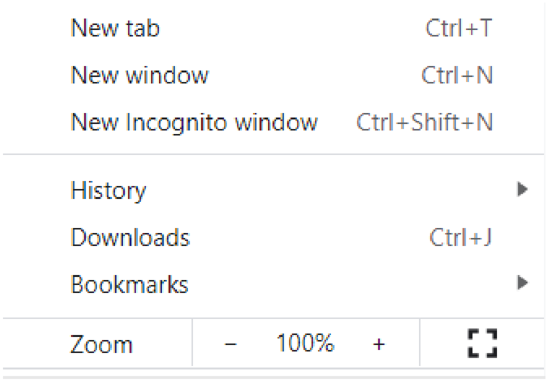
Originating Lead	test2 test2
Last Campaign Date	---
Marketing Materials	Send

DESCRIPTION

CONTACT PREFERENCES

Contact Method	Any
Email	Allow
Follow Email	Allow
Bulk Email	Allow

Every now and then, there would be a form that does not look well on your screen. You might try using browser's zoom capability to make it reflow into a better version of itself:



Of course, there is a limit to how much you can rely on the “zoom” feature, since, at some point, everything might become just too small to continue working comfortably.

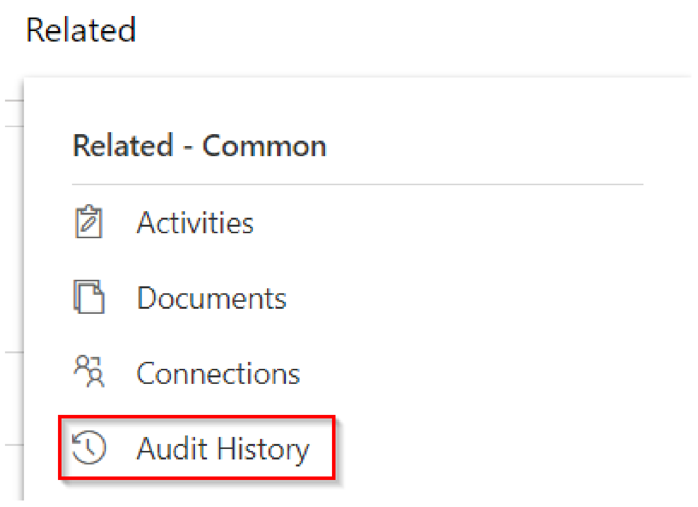
9. History tracking and audit capabilities

The question of tracking and audit may come from just about anyone working with model-driven applications, and it can be for different reasons.

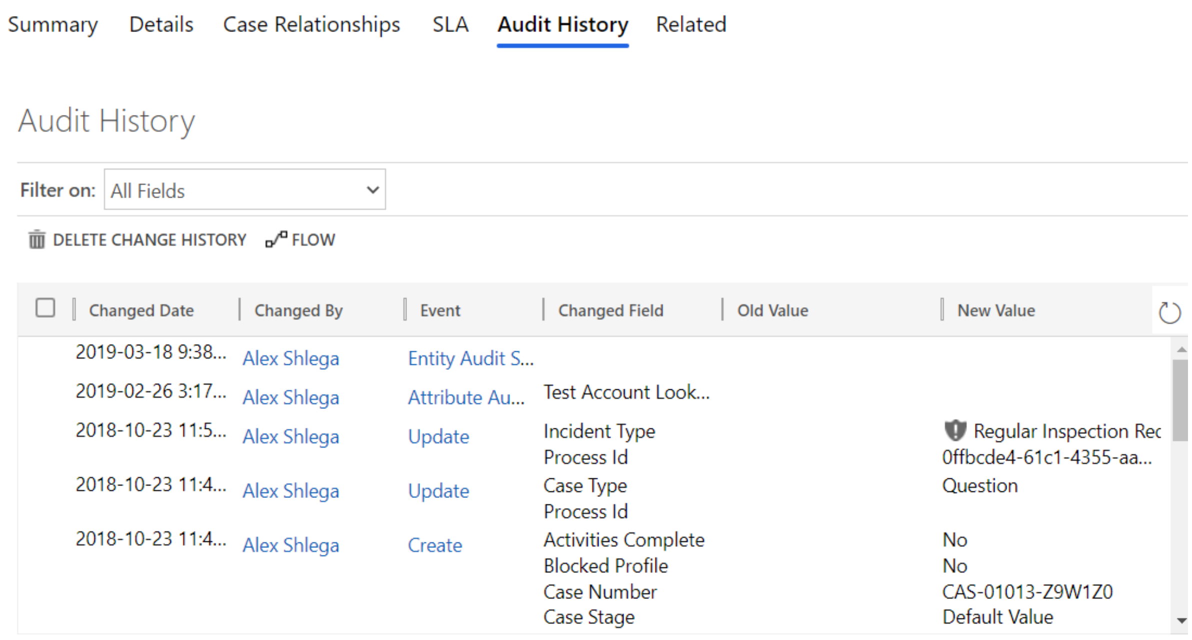
Therefore, it may be helpful to understand the basics of tacking and audit, and here is how it goes:

- There are "Created By", "Created On", "Modified By", "Modified On" columns in each table which provide basic history tracking functionality. Created By and Created On will be set once a record is created. Modified By and Modified On will be updated every time a record is updated (important: these two columns do not provide historical log - they are meant to maintain "modified on/by" details for the most recent update of the record)
- There are, also, audit logs. Auditing has to be enabled in the environment by the System Administrators. Also, it has to be enabled for the specific table, and for the specific columns in that table

Users with appropriate security roles will be able to see audit history in the environments. For example, when looking at the record form, "Audit History" for that record will be available as one of the items under "Related" dropdown:



Here is an example of audit history:



Audit History

Filter on: All Fields

Changed Date	Changed By	Event	Changed Field	Old Value	New Value
2019-03-18 9:38...	Alex Shlega	Entity Audit S...			
2019-02-26 3:17...	Alex Shlega	Attribute Au...	Test Account Look...		
2018-10-23 11:5...	Alex Shlega	Update	Incident Type Process Id		Regular Inspection Rec 0ffbcde4-61c1-4355-aa...
2018-10-23 11:4...	Alex Shlega	Update	Case Type Process Id		Question
2018-10-23 11:4...	Alex Shlega	Create	Activities Complete Blocked Profile Case Number Case Stage		No No CAS-01013-Z9W1Z0 Default Value

It’s important to keep in mind that Audit History is not meant for historical reporting - it is meant for troubleshooting and/or for figuring out what and when went wrong with the data.

10. Coming soon

- Ownership and assign operation
- Charts
- Dashboards
- Advanced Find
- Reporting
- Export to excel
- Sharepoint & Email integration (from the app user perspective)
- Word templates (from the app user perspective)

11. Live Course Delivery

Do you like it so far? That's great! However, having reading or video material around is awesome, but nothing will ever replace a live training targeted to a specific team/auditory.

If you are looking for the Power Platform training to be delivered to your team, please reach out to info@itaintboring.com, and we will see what and when can be done.