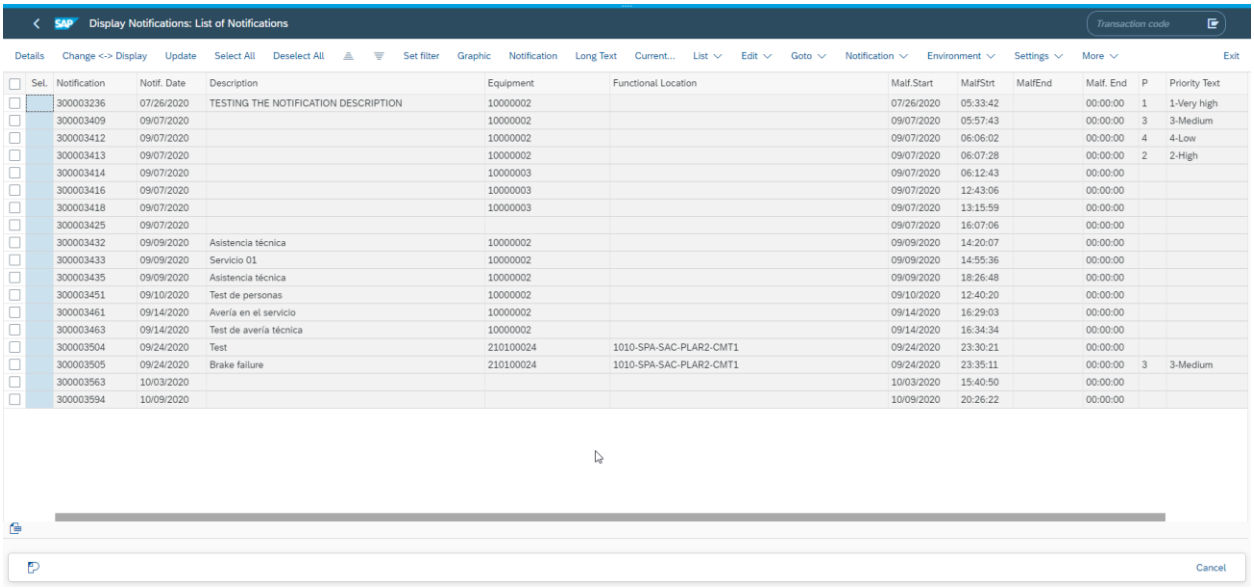
In this session, we will transform plant maintenance transactions IW29 and IW53, making them appear like Fiori applications. The goal is to provide the look, feel and main functionality of such an application, down to the responsive features and formatting which provides a much more delightful and modern experience than the original SAP GUI transactions. Simplifications to screens will tailor the backend transactions to the needs of our specific user group.

Since our audience mostly use mobile devices, we will focus on tablet users first.

First task: Improving the look and usability of the Notification List

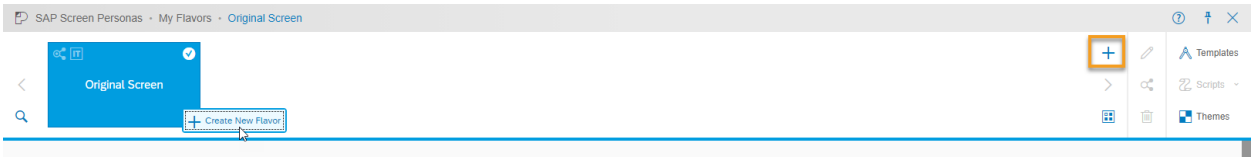
Let's start with IW29 which allows selection of notifications. Our users are interested in service notifications, so enter S* as the Notification Type selection and hit 'Execute' to get the result list:



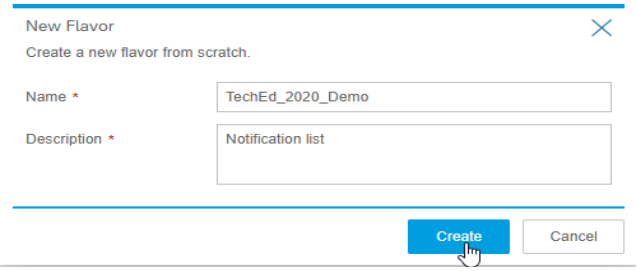
Sel.	Notification	Notif. Date	Description	Equipment	Functional Location	Malf.Start	MalfStrt	MalfEnd	Malf. End	P	Priority Text
	300003236	07/26/2020	TESTING THE NOTIFICATION DESCRIPTION	10000002		07/26/2020	05:33:42		00:00:00	1	1-Very high
	300003409	09/07/2020		10000002		09/07/2020	05:57:43		00:00:00	3	3-Medium
	300003412	09/07/2020		10000002		09/07/2020	06:06:02		00:00:00	4	4-Low
	300003413	09/07/2020		10000002		09/07/2020	06:07:28		00:00:00	2	2-High
	300003414	09/07/2020		10000003		09/07/2020	06:12:43		00:00:00		
	300003416	09/07/2020		10000003		09/07/2020	12:43:06		00:00:00		
	300003418	09/07/2020		10000003		09/07/2020	13:15:59		00:00:00		
	300003425	09/07/2020				09/07/2020	16:07:06		00:00:00		
	300003432	09/09/2020	Asistencia técnica	10000002		09/09/2020	14:20:07		00:00:00		
	300003433	09/09/2020	Servicio 01	10000002		09/09/2020	14:55:36		00:00:00		
	300003435	09/09/2020	Asistencia técnica	10000002		09/09/2020	18:26:48		00:00:00		
	300003451	09/10/2020	Test de personas	10000002		09/10/2020	12:40:20		00:00:00		
	300003461	09/14/2020	Avería en el servicio	10000002		09/14/2020	16:29:03		00:00:00		
	300003463	09/14/2020	Test de avería técnica	10000002		09/14/2020	16:34:34		00:00:00		
	300003504	09/24/2020	Test	210100024	1010-SPA-SAC-PLAR2-CMT1	09/24/2020	23:30:21		00:00:00		
	300003505	09/24/2020	Brake failure	210100024	1010-SPA-SAC-PLAR2-CMT1	09/24/2020	23:35:11		00:00:00	3	3-Medium
	300003563	10/03/2020				10/03/2020	15:40:50		00:00:00		
	300003594	10/09/2020				10/09/2020	20:26:22		00:00:00		

This is presented as a grid with many columns, from which a lot are not relevant for our target user group. We could change this list with regular flavor editing means, but our goal is to provide the result in a format resembling a Fiori application. In other words, we want to transform it into a SAPUI5 table.

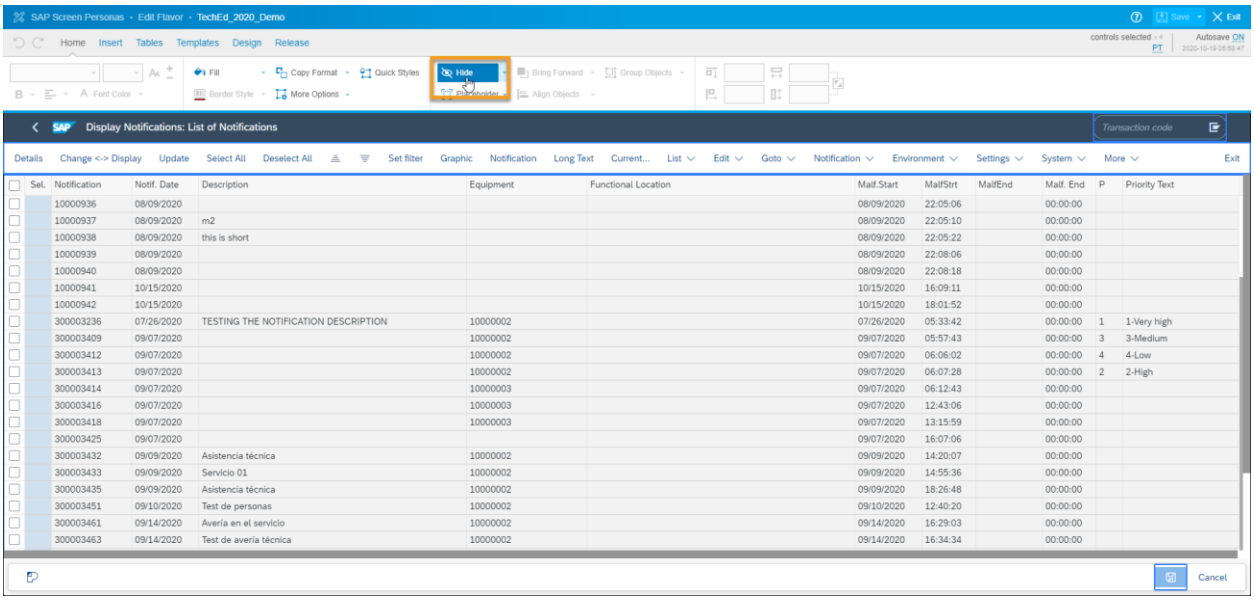
To do this, let's create a flavor first. Open the Personas toolbar by hovering over the top blue line and clicking on the Personas "P" in the middle, or by clicking on the "P" in the lower left corner. Right-click on the 'Original Screen' tile and select 'Create New Flavor'. Alternatively, you can also click the plus sign on the right side:



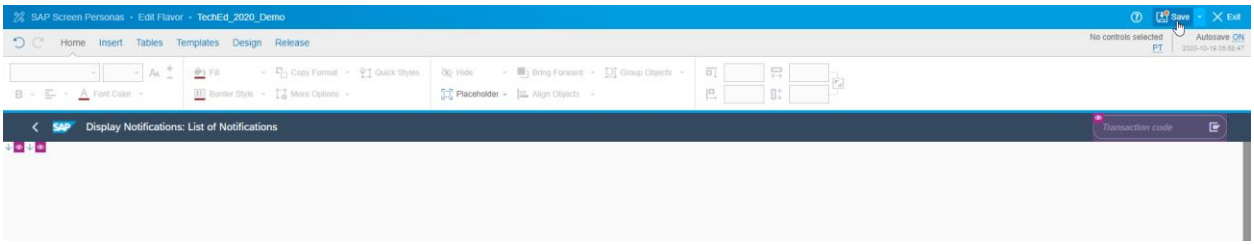
Specify a flavor name and description (can be anything, doesn't have to be unique), then hit 'Create':



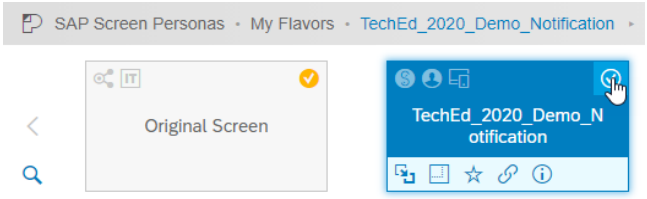
First, let's remove the controls that we will replace or won't need. Multi-select (hold down Ctrl, then left click) the 'Transaction code' command field, menu bar, the whole grid and the Save button in the footer, then click 'Hide' in the Home tab:



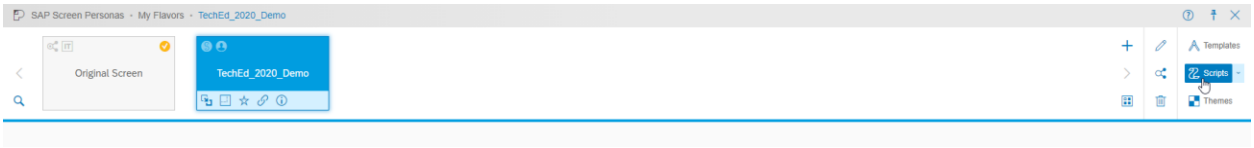
Hit 'Save', then exit the editor.



Set this new flavor as your default by clicking on the check mark in the upper right corner of the flavor tile:



Open the script editor:



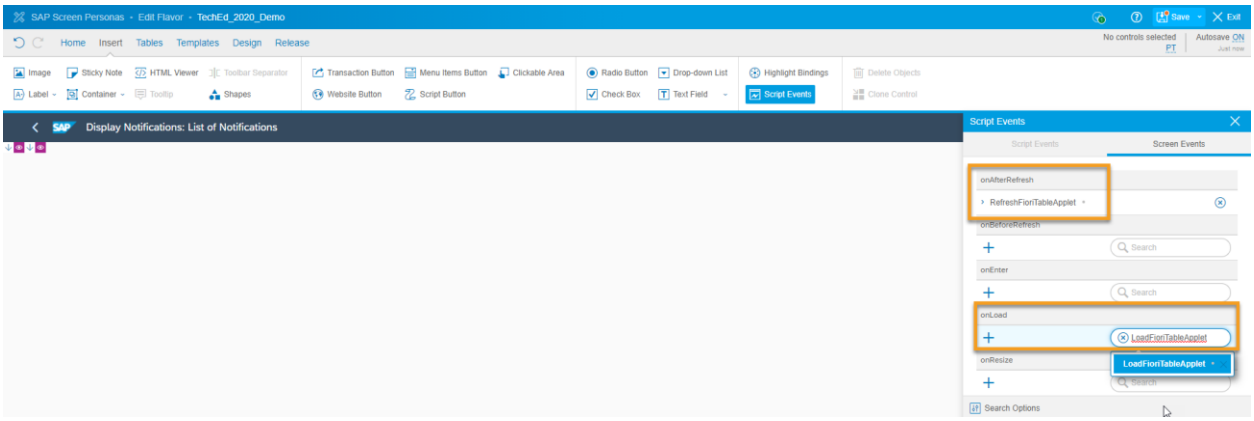
To change the result list to a SAPUI5 table, we will use a UI5 applet that is already deployed on the backend ABAP system. The applet takes the data from this table on the screen and replaces it with a SAPUI5 table. Our job is to link the applet to the result screen.

The necessary scripting is a little bit more involved than the usual tasks, so we provided two scripts for you to reuse. Expand the 'Global Scripts' area on the left.

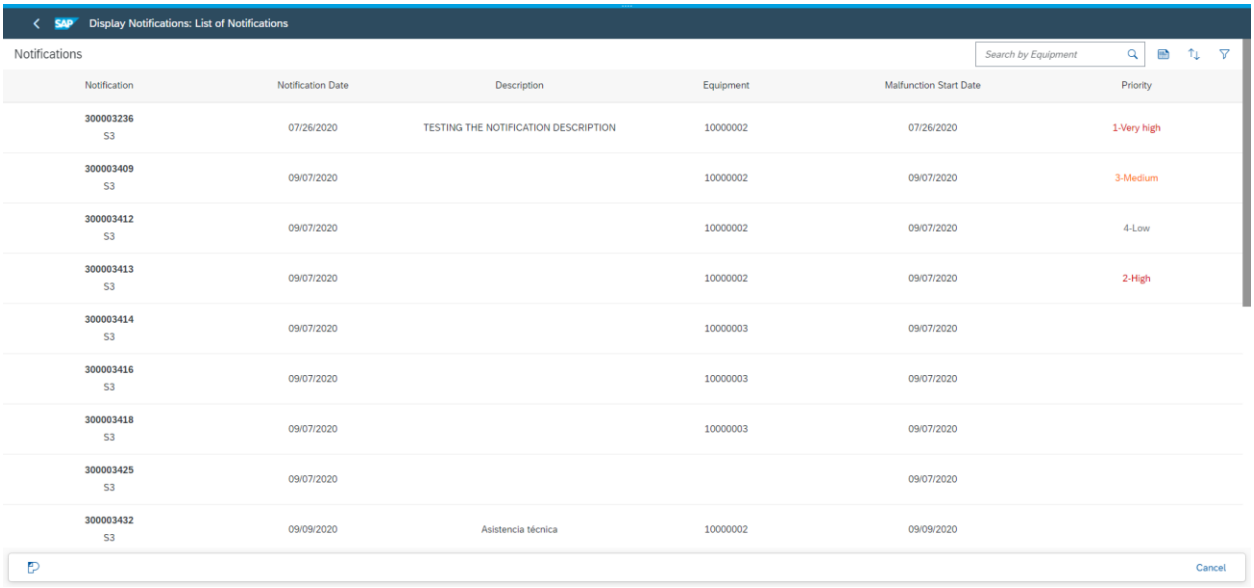
The first script is called 'LoadFioriTableApplet' and as its name says, this will load the applet, then initialize it by reading the data from the result grid and storing it for later use. It also executes the second script, which displays the transformed table. We need to attach this to the screen's *onLoad* event.

The second script is called 'RefreshFioriTableApplet'. This reads the data stored by the initialization script and takes care of rendering the table via the SAPUI5 applet. This will be linked to the screen's *onAfterRefresh* event.

Let's take care of these bindings. Leave the script editor and go back to the flavor editor. In the Insert tab, click on 'Script Events', then for the *onAfterRefresh* event, start typing Refresh… and select the **RefreshFioriTableApplet** script by clicking its name. Similarly, for *onLoad* select the script **LoadFioriTableApplet**:

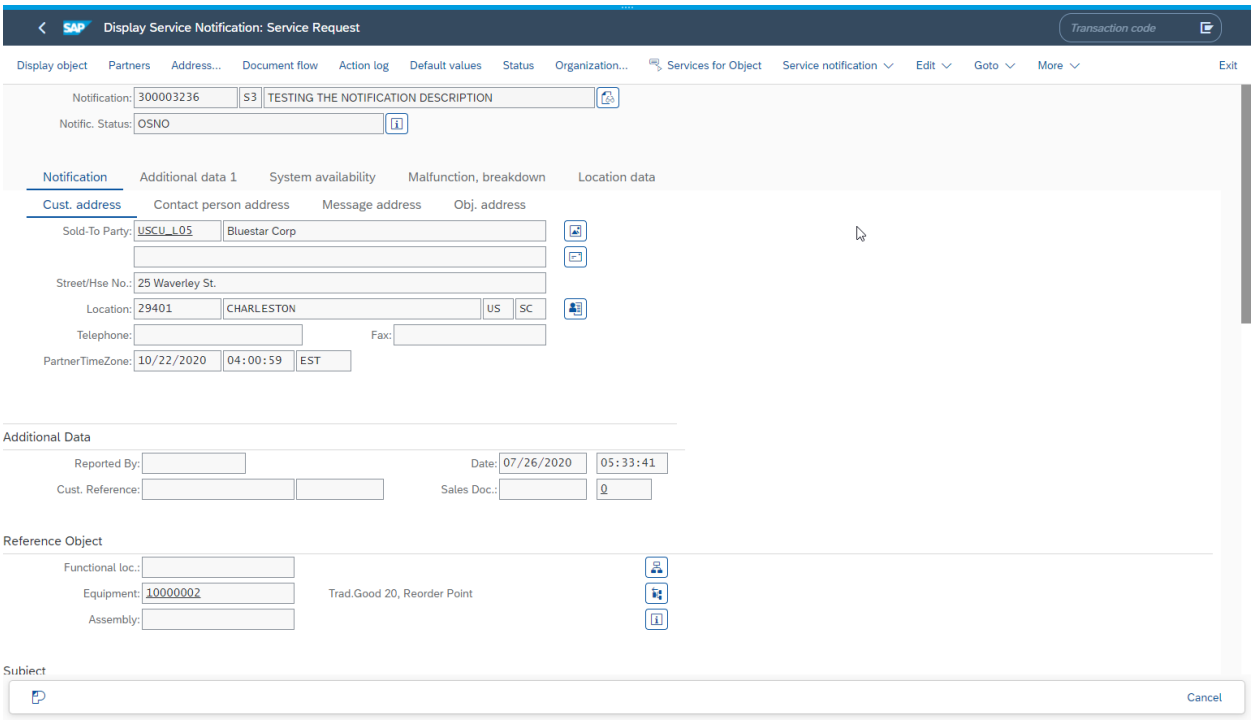


Save the flavor and exit the editor. Don't be alarmed by the blank screen. Since now the grid is hidden and our applet is not initialized yet, this is expected. Return to the selection screen by clicking the 'Back' button in the upper left corner, then re-run the report by hitting 'Execute'. Now the scripts will run, and the result is presented as a beautiful SAPUI5 table, complete with sorting, filtering and search capabilities. It is also responsive, reacting to window size changes, resembling the look and behavior of a Fiori list report:



The SAPUI5 applet is also coded to respond to line selection. When clicking on a row in the table, it will run transaction IW53, displaying the selected notification.

Of course, that screen looks like a regular SAP GUI transaction rendered by Slipstream Engine, so we got to do something about the look and feel of it to match our notification list. This is what we start with:



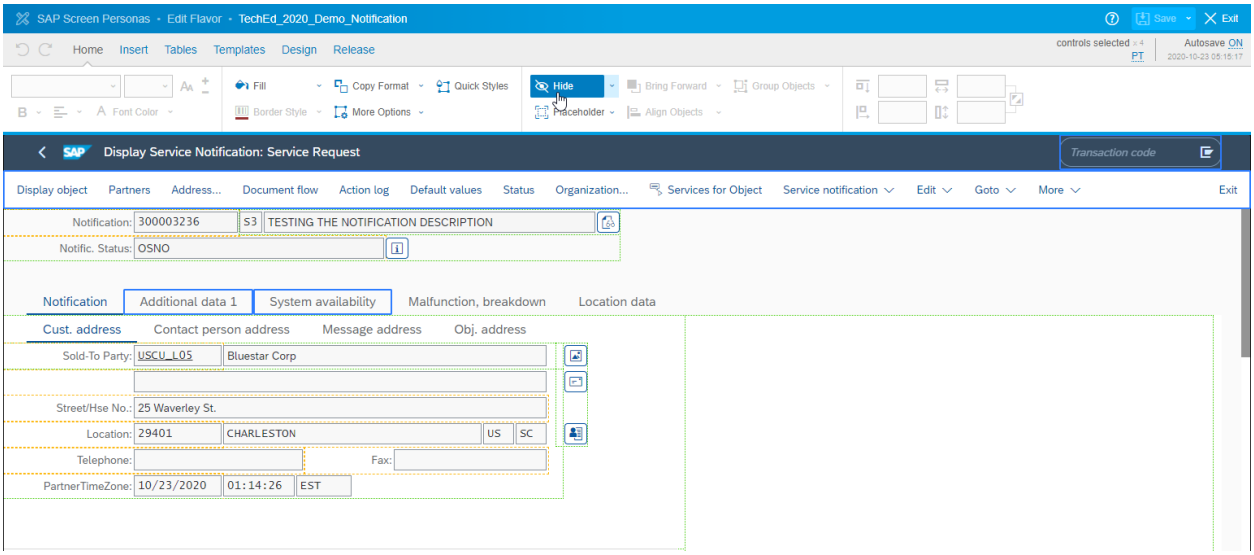
There are plenty of opportunities for improvement, so let’s do that next.

Second task: Turn the Display Notification screen into a Fiori object page

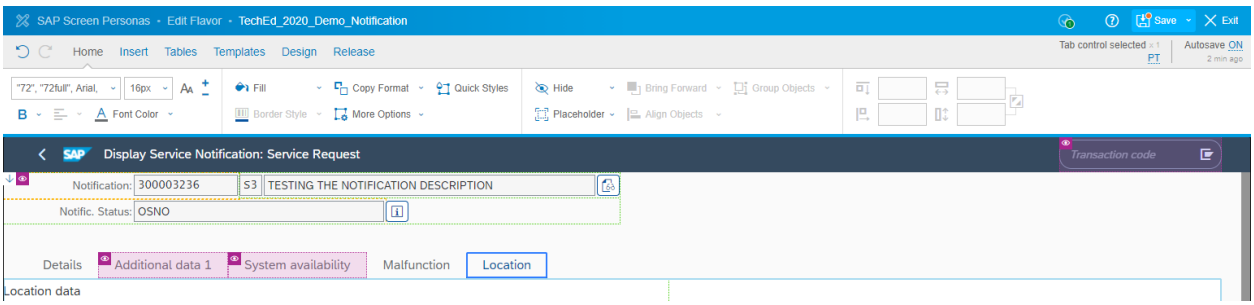
The Display Notification screen shown above is rather busy, and we want to provide our users a simpler, better looking application that adheres to the Fiori design guidelines.

During our Design Thinking sessions with our users, we concluded that they are interested in information which can be grouped into general details, the malfunction data and the location of the equipment the notification is about.

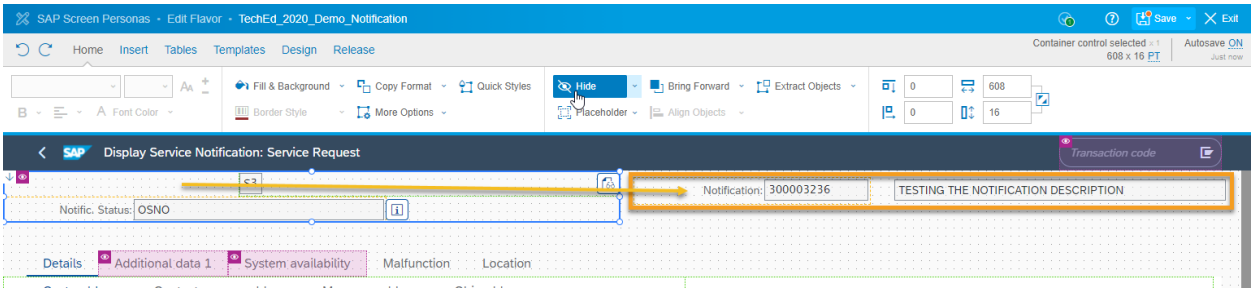
Create a new flavor for this transaction. You did this earlier with IW29, so follow the same steps. As the first change, let's hide the 'Transaction code' field and the menu bar. We can also hide the 'Additional data 1' and 'System availability' tabs, because our users don't need them.



Let's rename the three remaining tabs. Double-clicking its header switches to a tab, and by repeated double-click we can change the label to Details, Malfunction and Location, respectively.

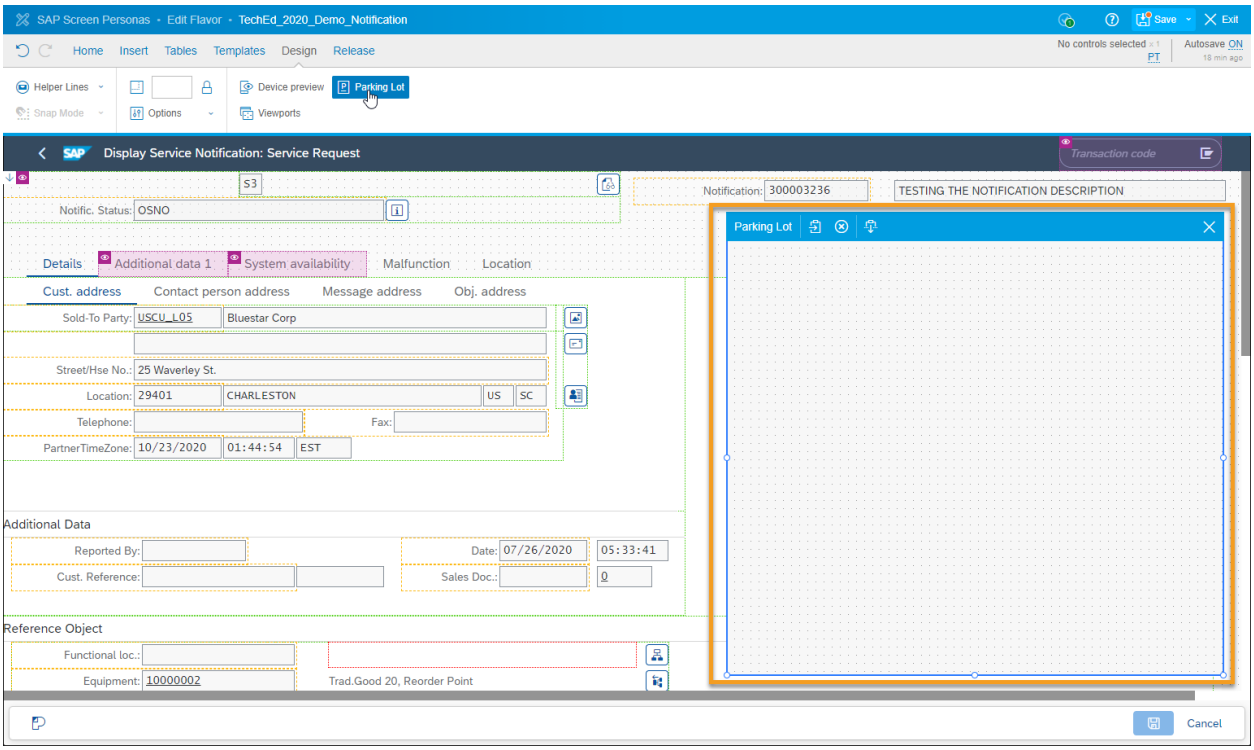


In the header area, we need the notification number with its label and the short description, everything else should be hidden. Let's grab these three controls and drag them aside on the right, where there is some free space. Then, we can hide the container for the other header fields:



Switch back to the 'Details' tab. From here, we will need a few objects. When simplifying tabs, we can once again drag them to an empty area outside the tab strip before moving them to their correct place. However, in our case this is a little more difficult because the tab strip is wide and our screen may not be big enough, resulting in a lot of horizontal scrolling. In this scenario, a great tool to collect these in an intermediary location is the so-called 'Parking Lot' which is something like a clipboard where we can temporarily store controls, then place them where we want them later.

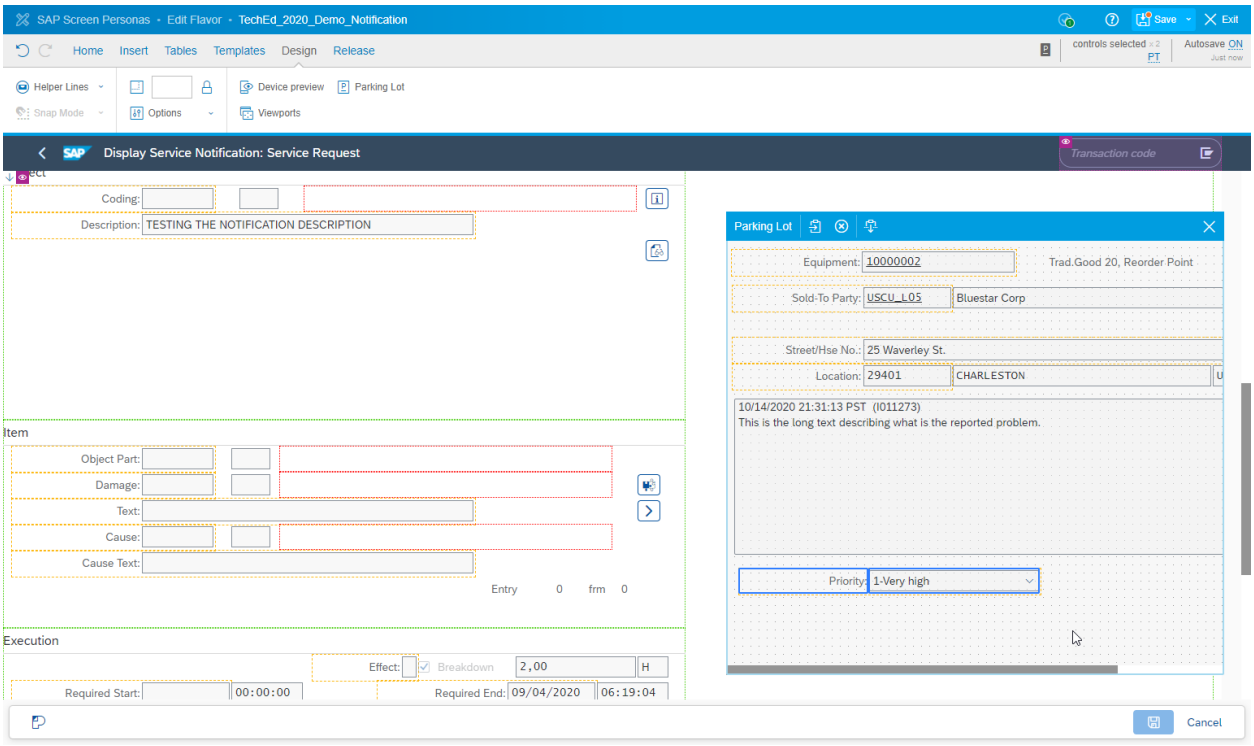
In the Personas toolbar, go to tab 'Design', then click 'Parking Lot'. An additional window opens, which we can move or resize as necessary, and move objects to it from the screen:



Let's move the following to the Parking Lot:

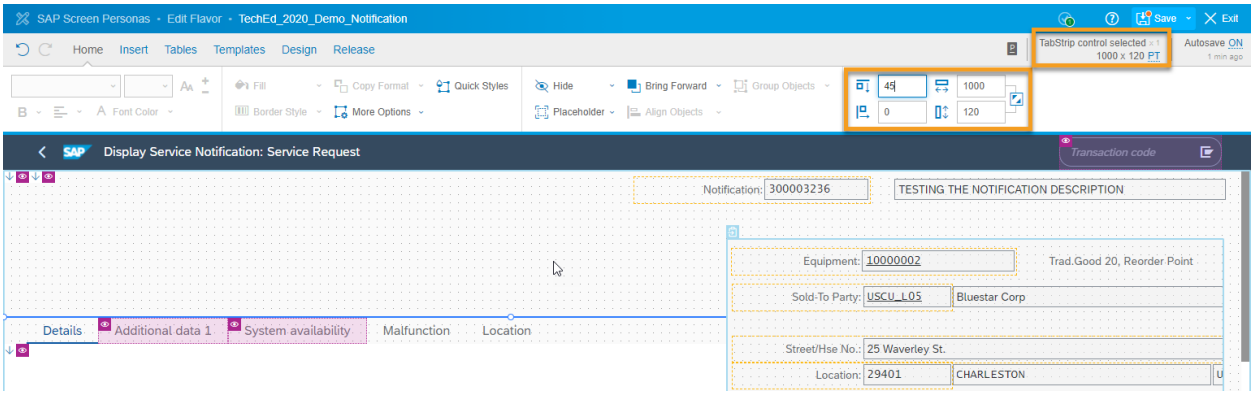
- The Sold-To Party label, ID, name
- Street/House no. label and field
- Location label, ZIP code, city, country, state
- From the 'Reference Object' group box:
    - Functional loc. label, ID and description
    - Equipment label, number and description

- From the ‘Subject’ group box, the long text
- From the ‘Execution’ group box, the Priority label and dropdown



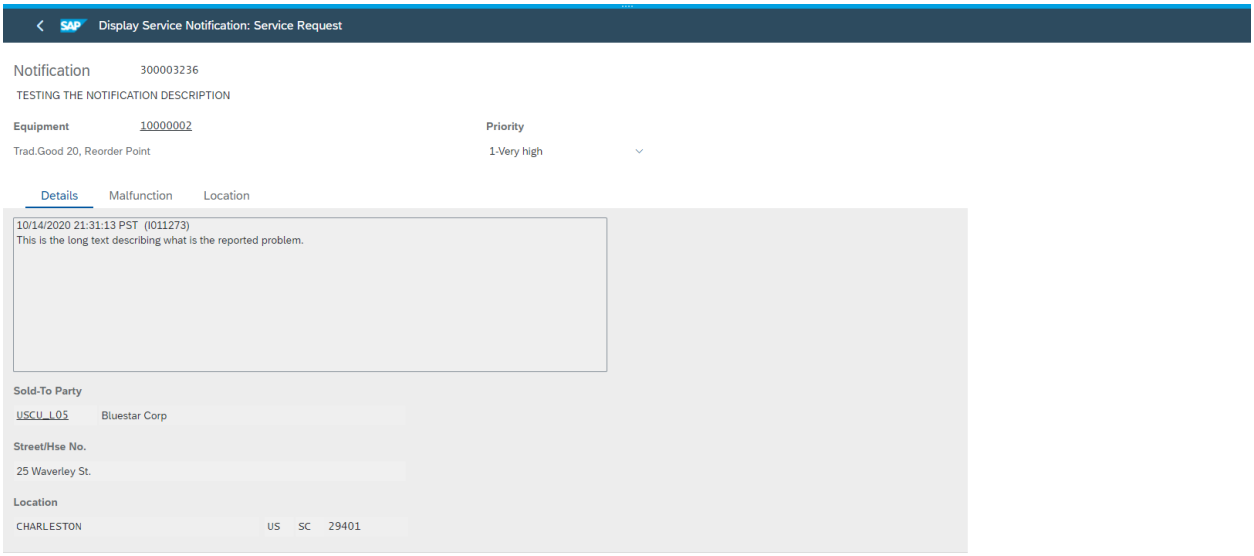
Now, let’s go ahead and remove all remaining controls in this tab so we end up with an empty ‘Details’ tab. Simply (multi)select those container controls / group boxes and hide them, along with their content.

As the next step, resize and move down the complete tab strip control to make some more space for the header. Ensure that you select the correct object. The recommended values are:



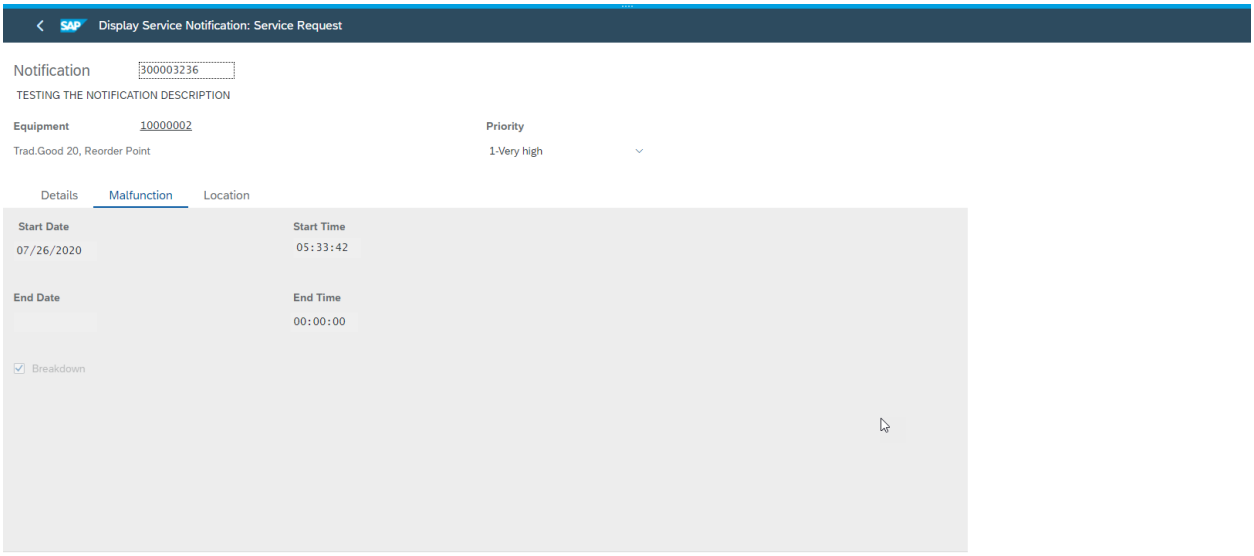
It’s time to move the ‘parked’ controls to their proper location. The equipment number and priority go to the header area above the tab strip. The others should be moved to the ‘Details’ tab, except for the Functional Location, which we will use later. Don’t forget, you can drag and resize the Parking Lot as necessary, if it’s in the way or if something in it is not visible.

Relocate the header fields and format everything so the screen layout looks like the following screen shot.



Field labels should be bold (use multi-select here to change the font decoration). To get the gray tab background color, select the *ScrollContainer* control in the tab, then 'Fill and Background' in the Personas 'Home' tab. Go to 'Mixer' and enter the hex code EDEDED in the # field.

With this, the first tab is done. Let's move to the 'Malfunction' tab. This is simpler; all we need is reorganize and reformat the controls like this:

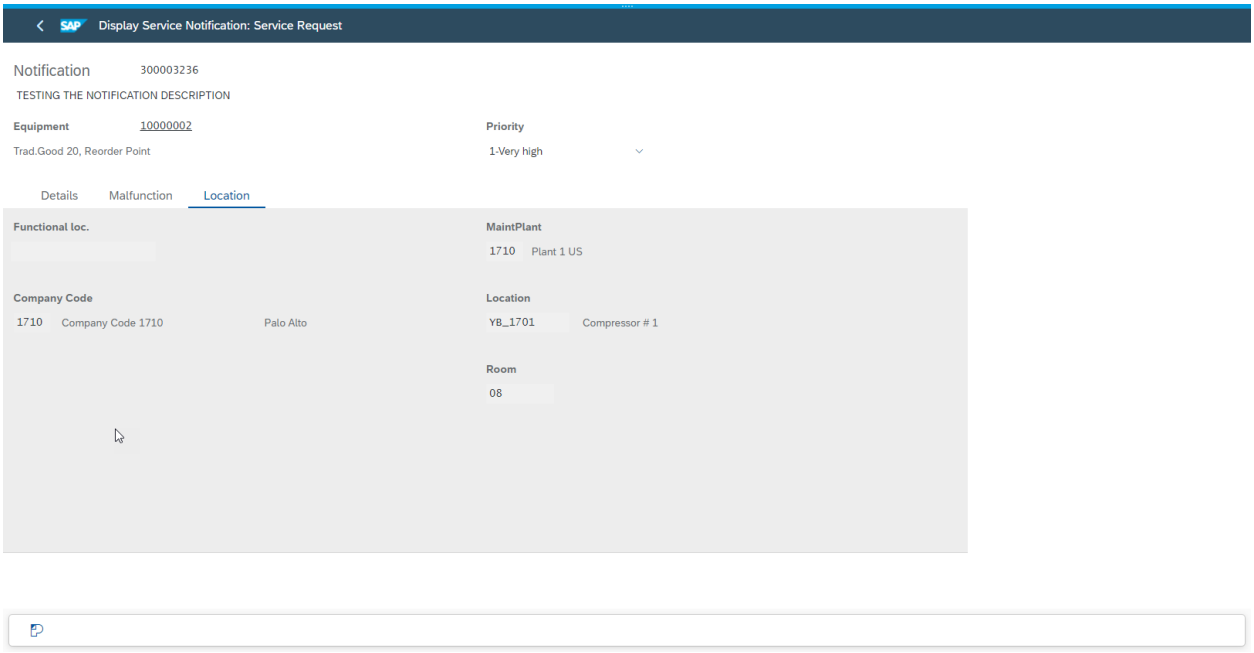


Steps:

- Change field labels (by double-clicking)
- Move the necessary controls out of their group box containers

- Hide the group boxes
- Arrange the objects according to the screen shot above
- Set the tab background color like earlier

Last editing step is the ‘Location’ tab. Here, the process is like with the ‘Malfunction’ tab, however we still have the ‘Functional Location’ field and description sitting in the Parking Lot. This is when they should be moved to the ‘Location’ tab, so the final look of this tab is as follows:

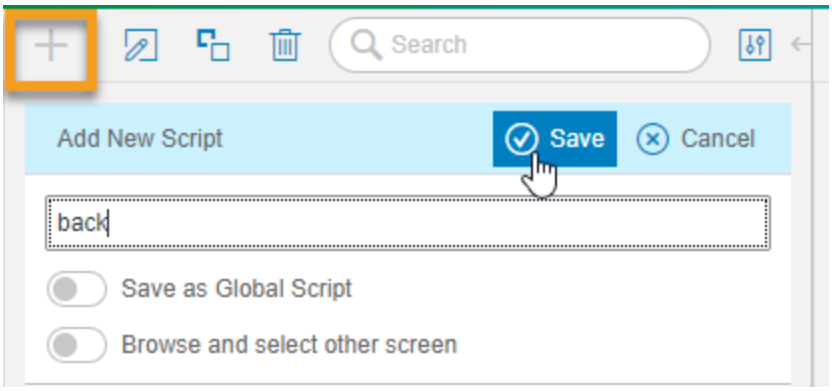


With this, the layout editing part of our flavor is complete. We significantly reduced complexity of this screen and gave it the look of a Fiori application.

There is one problem remaining though. We would certainly want to be able to return to our notification list and select another document for display. With the original form of IW29 going to IW53, this is provided automatically. However, since we replaced the list grid in IW29 with the SAPUI5 table, this is not working anymore. When we selected a document, a script called IW53 as a new transaction, passed on the selected notification number and skipped the first screen. This means that if we now click the ‘Back’ button in IW53, we end up on the selection screen of that transaction, instead of returning to the IW29 list. We need to fix this.

Essentially, instead of the normal ‘Back’ button functionality, we need to re-run transaction IW29 and get past its selection screen. This is simple to do with a script, which we will attach to the standard ‘Back’ button.

Exit the flavor editor and open the script editor. Create a new script by pressing the  sign and entering a name.



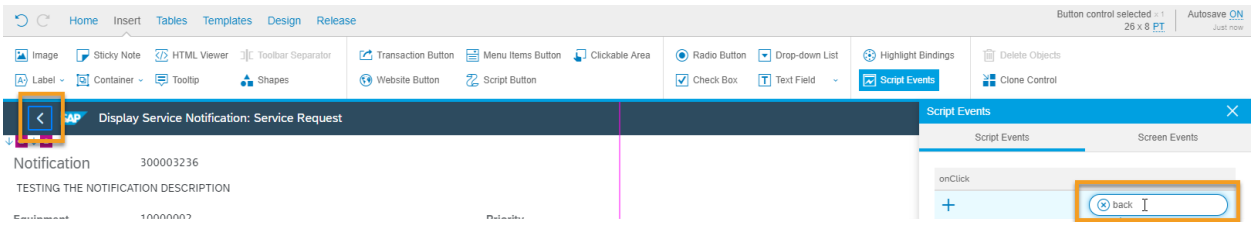
You could use recording to capture the steps, or here is the script:

```
// Start IW29
session.findById("wnd[0]/tbar[0]/okcd").text = "/nIW29";
session.findById("wnd[0]").sendVKey(0);

// Enter the notification type selection
session.findById("wnd[0]/usr/ctxtQMART-LOW").text = "S*";

// Go to the list (execute selection by pressing F8)
session.findById("wnd[0]/tbar[1]/btn[8]").press();

// Suppress standard 'Back' button functionality
return true;
```

The last step is necessary, since we want to override the standard ‘Back’ button functionality with our script, and this will do exactly that.

Save the script, exit the script editor and go back to the flavor editor. Now we need to link this script to the ‘Back’ button.

Select the  ‘Back’ button in the upper left corner, switch to the ‘Insert’ tab in the Personas toolbar and click ‘Script Events’. Attach the new ‘back’ script to this event.



Save and exit the Editor. Once again, set your new flavor as your default for this transaction by clicking the check mark in the upper right corner of the flavor tile.

Now, if you click the ‘Back’ button, it will take you back to the IW29 list. Since the SAPUI5 app’s context was saved, it knows exactly where you were when selecting the notification, so it can restore the list to that state.

Third task: “Display Notification” flavor for phone users

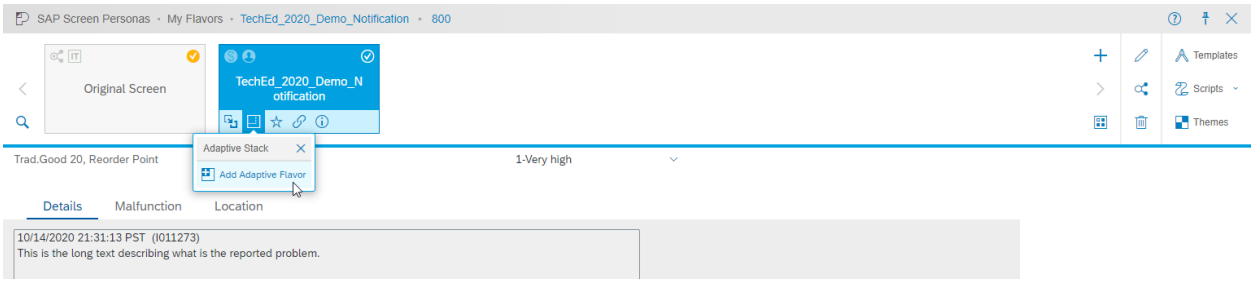
Now that we handled users with tablets (or a desktop), let’s cater to those who are running this process on a phone. The screen is smaller, and its orientation is portrait rather than landscape. The new IW29 look with the SAPUI5 table is automatically responsive, rearranging the layout as necessary. This is not happening with our IW53 flavor, so we need to take care of this with a so-called adaptive flavor, designed for the phone screen. These can be considered as children of a flavor, which are automatically picked based on the actual screen width. We can have as many such adaptive flavors as necessary, for different screen sizes / devices. Our targeted phone model will be an iPhone 8.

For this to work, it is essential that each flavor in the adaptive stack has its minimum width set, including the parent flavor. So, let’s change our tablet (parent) flavor and set its width to 800. This means that this flavor will be selected if the screen size is at least this wide.

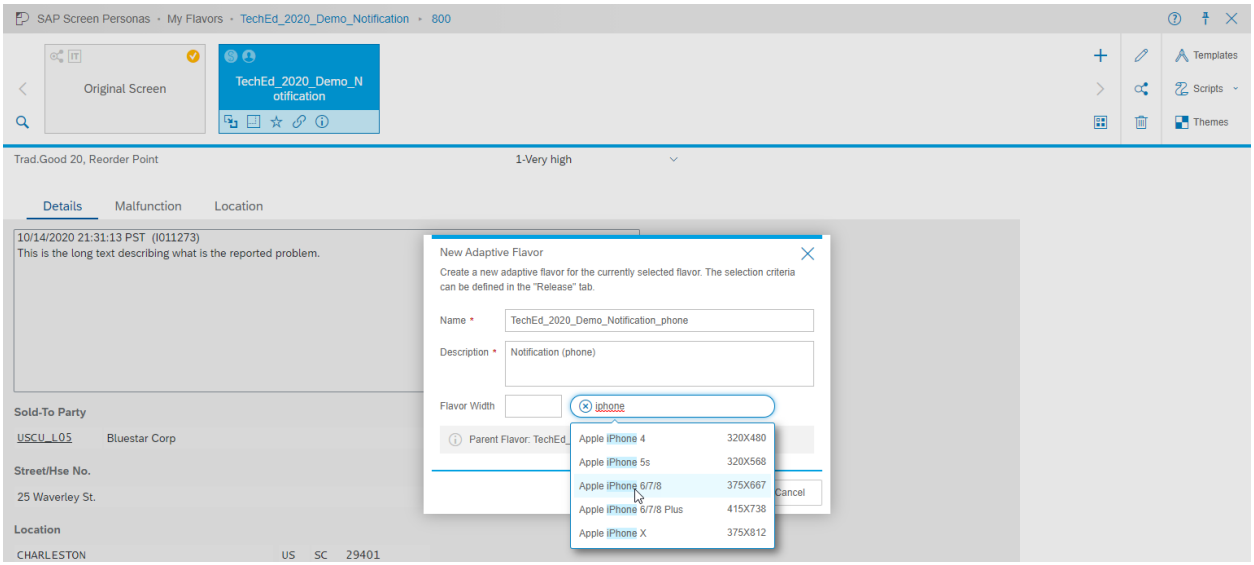
Open the flavor editor and go to the ‘Design’ tab. Here, enter 800 as the minimum width:

Save the change and exit the editor.

Click on the flavor tile, then select ‘Show Adaptive Flavors, then ‘Add Adaptive Flavor’.

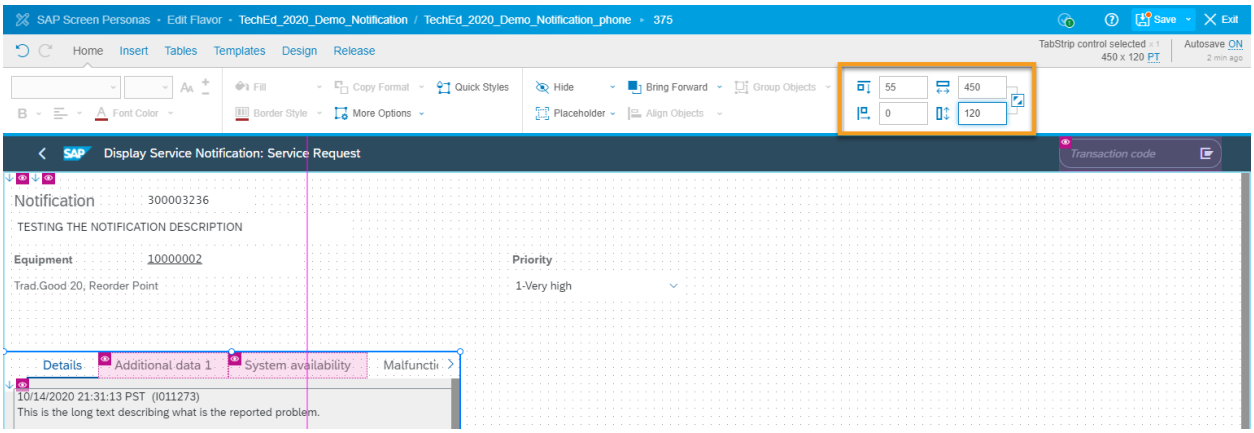


Remove the pre-selected width, enter ‘iphone’ in the device type field, and select iPhone 6/7/8 from the dropdown. Notice that the minimum width setting is also taken from this selection, so your new phone flavor is set up properly in the adaptive stack, and will be automatically selected if users runs the parent flavor on their device with a screen narrower than the parent flavor’s 800 width.

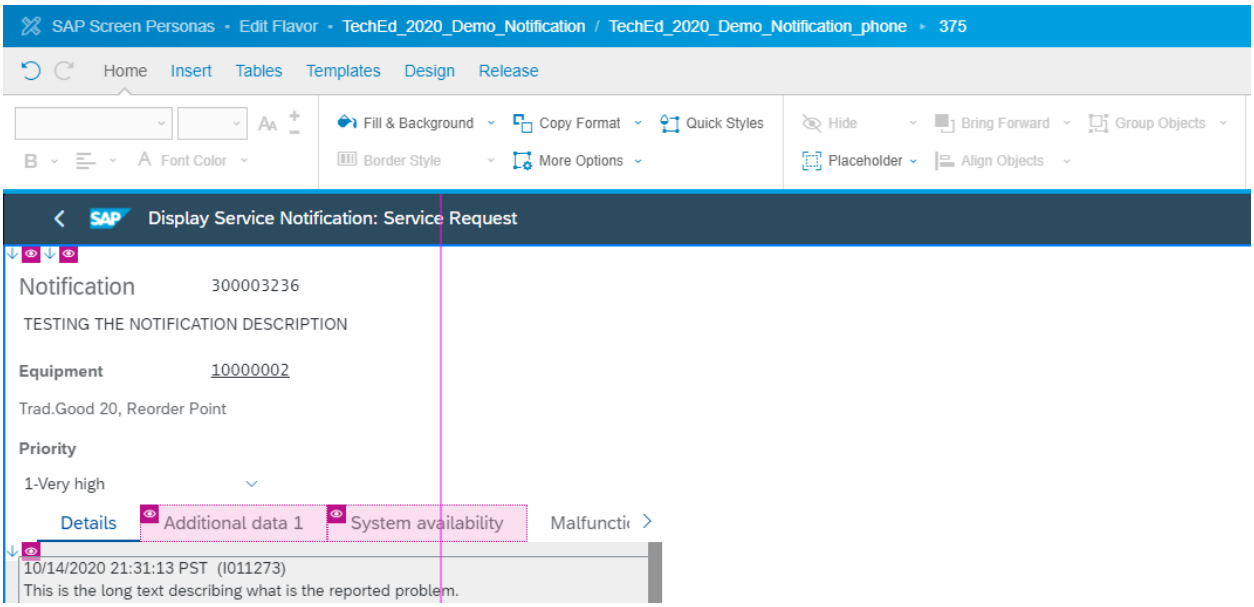


Your phone flavor inherits the settings from the parent tablet flavor. You need to rearrange the controls somewhat, so they fit the changed screen dimensions better.

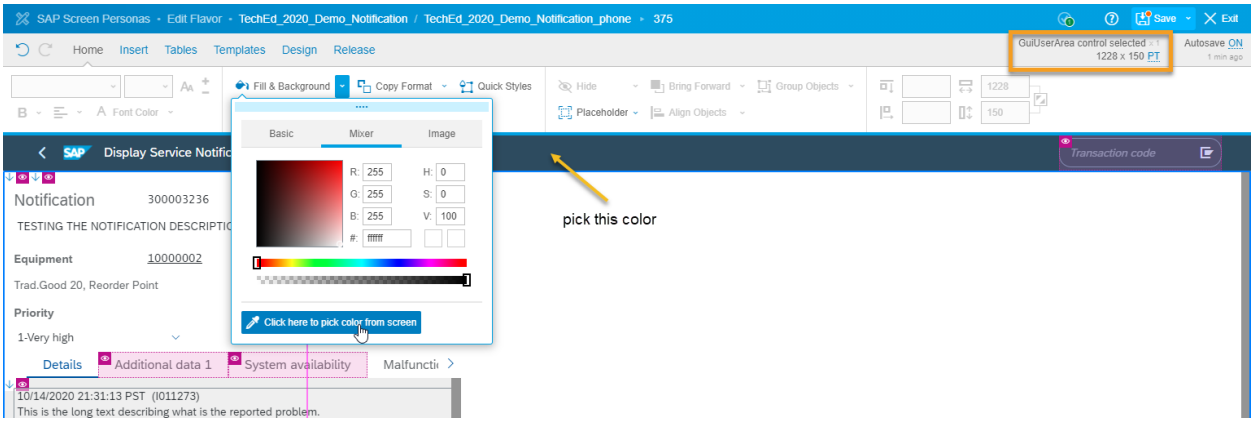
First, move the tab strip down so the Priority can fit above it, and make it narrower too. The recommended settings are:



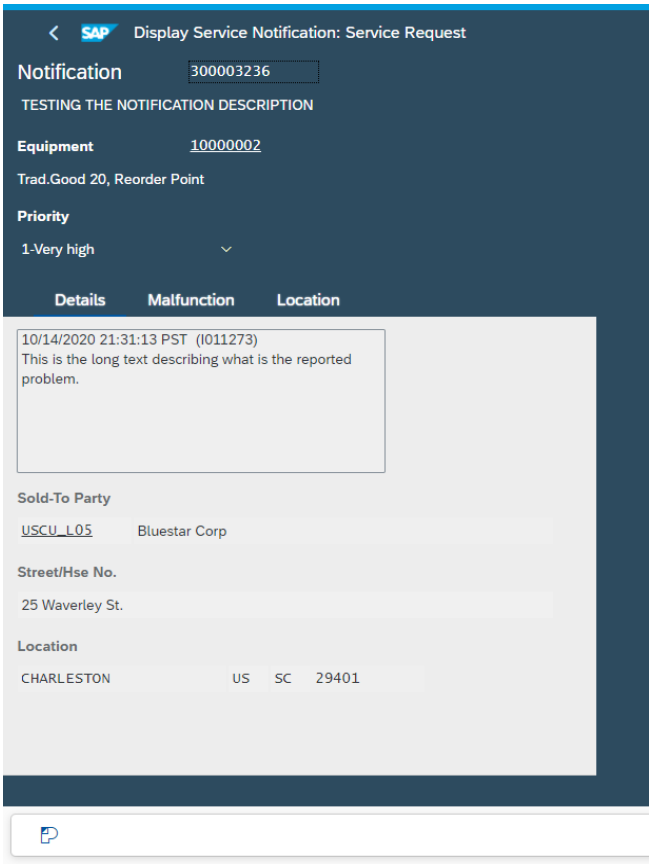
Now, move the Priority to the left, under Equipment:



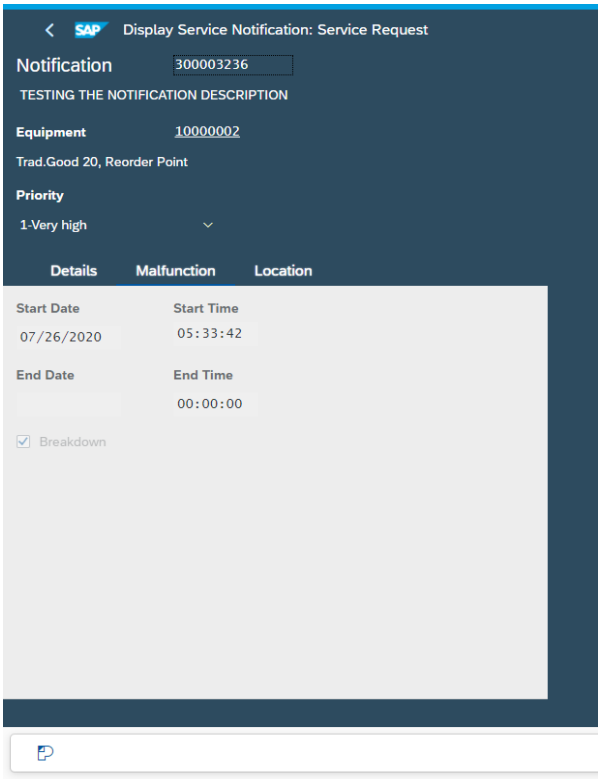
Let's change the header background color to match the standard title bar. Select the User Area by clicking outside the tab strip and not on any header field, then Fill & Background, Mixer and the color picker. Pick the title bar's color:

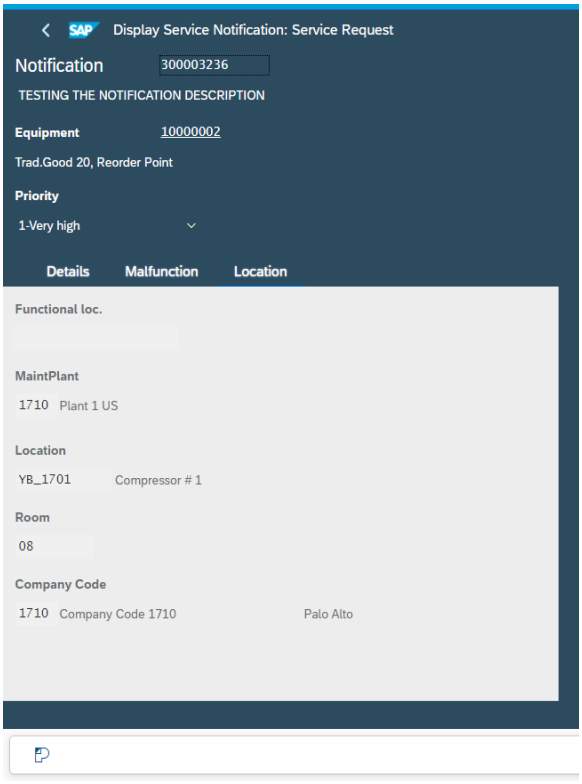


Then set the same fill color for all objects in the header, and their font color to white. The resulting layout should be like this:



Let's move on to the other two tabs. Do the same and rearrange the layout to get them look like this:



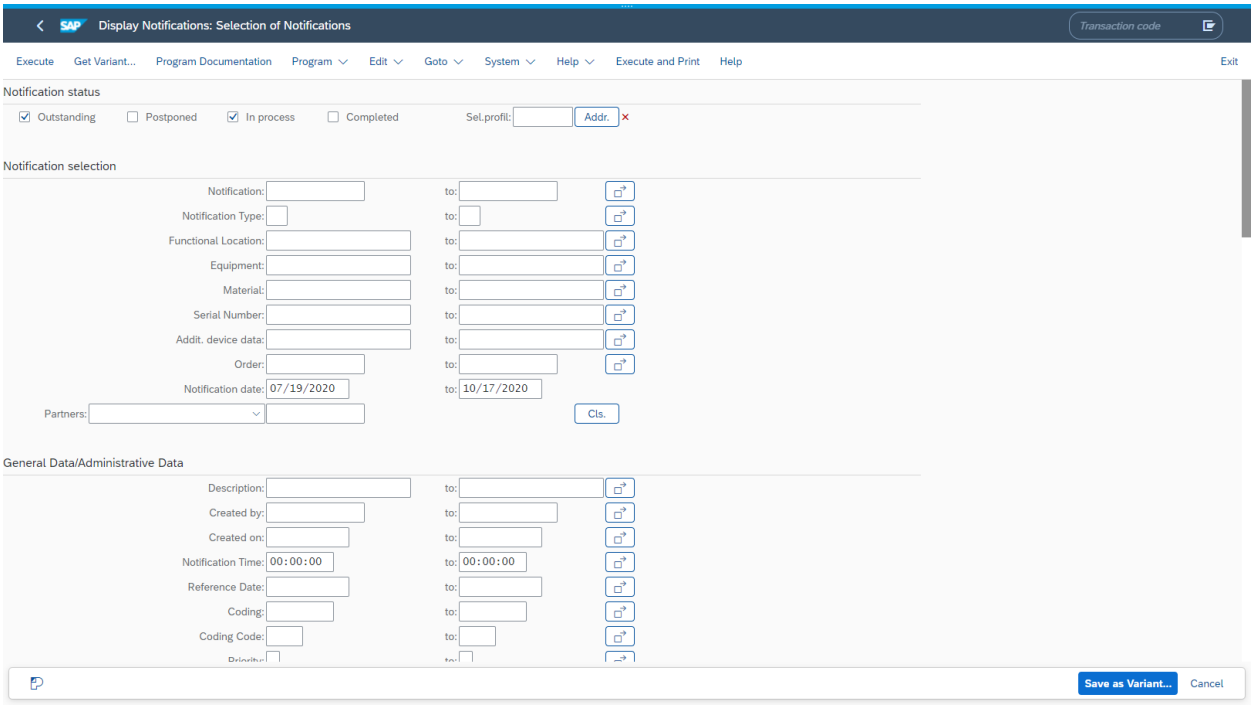


Now, you can test how the proper flavor is selected. Decrease the width of your browser window by dragging its vertical borders. Once you hit the defined width threshold, the flavor according to the current width will be picked.

You can test this on your phone too. Log on with your credentials via the same URL and run IW29, then select a notification from the list. Turning the screen orientation will adjust the layout according to the currently active width, both in the notification list and your notification flavor.

Fourth task: Adjust the selection screen of the Notification List

The selection screen of this transaction is very busy, with most selections irrelevant for our users:

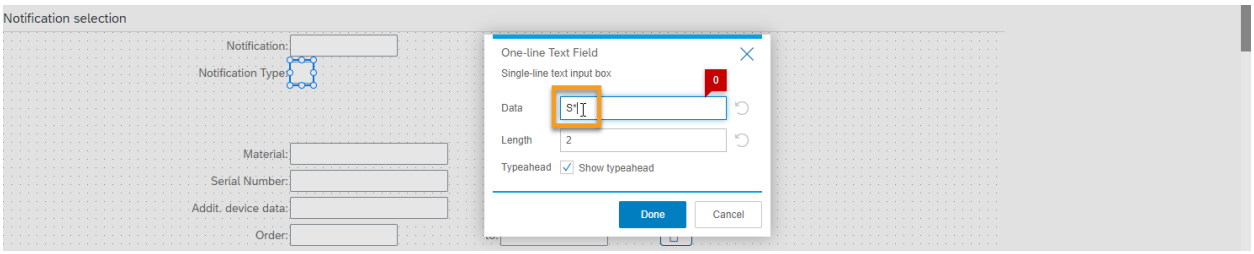


Besides, there are a few things we can improve on in the process:

- Since we only want to consider the service notifications, it makes sense to automate that selection
- While the SAPUI5 table does offer filtering capabilities, the notification date range selection is not part of this and that may be important for our users to change from the default
- Besides selecting per equipment, we also want to restrict our list by Functional Location

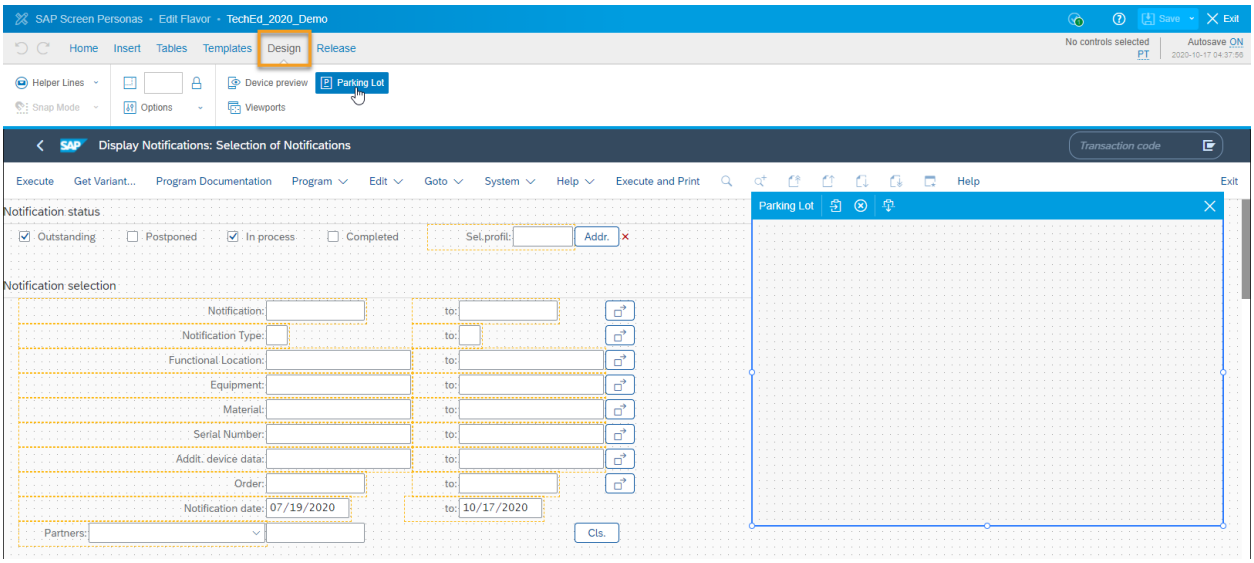
So, let's simplify it as the first step and remove everything we don't need.

First, set the Notification Type selection value by double-clicking the "from" value. This opens a popup where you can set the Notification Type value to S*:



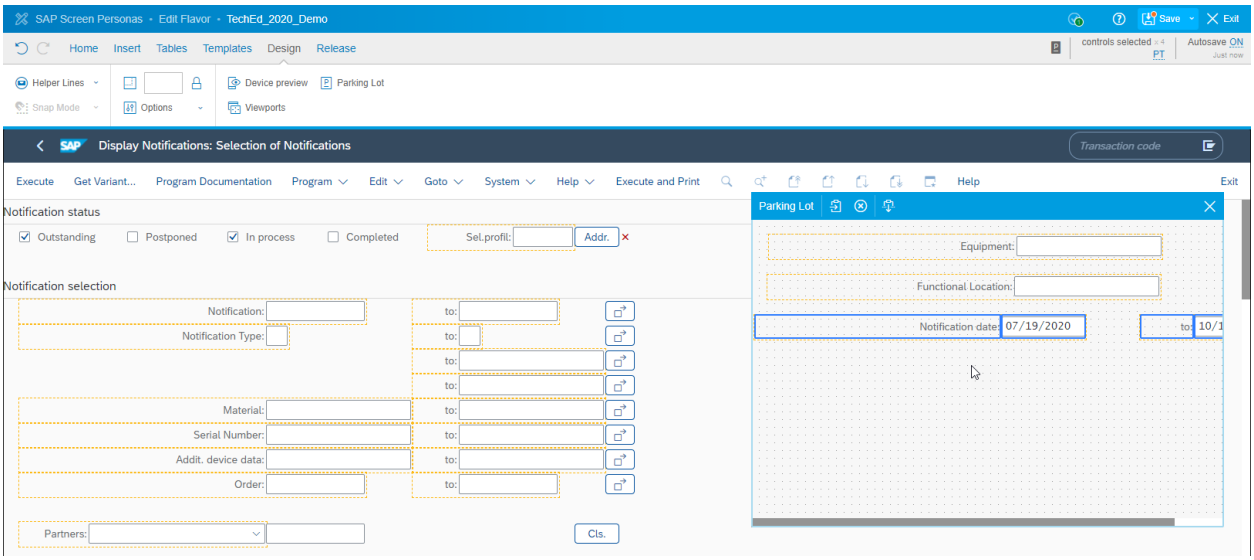
Hit 'Done', and now the selection is pre-set as part of the flavor.

Moving on to hide the unneeded objects, open the 'Parking Lot' in the Personas toolbar's 'Design' tab.

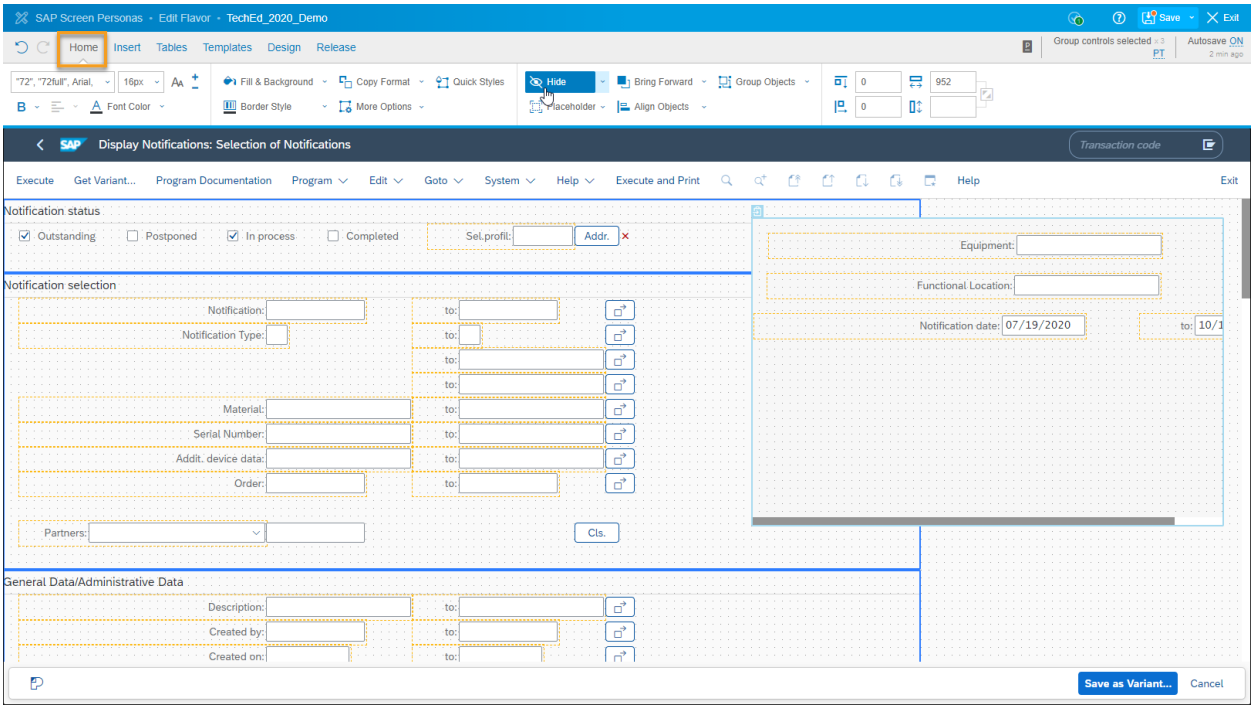


Let's move the labels and selection fields we want from the standard screen to the 'parking lot'. You can multi-select controls by holding the 'Ctrl' key and clicking them with the left mouse button, or even draw a box around multiple fields while holding 'Shift' and the left mouse button, then dragging the cursor. When you selected the fields you need, simply drag them to the parking lot.

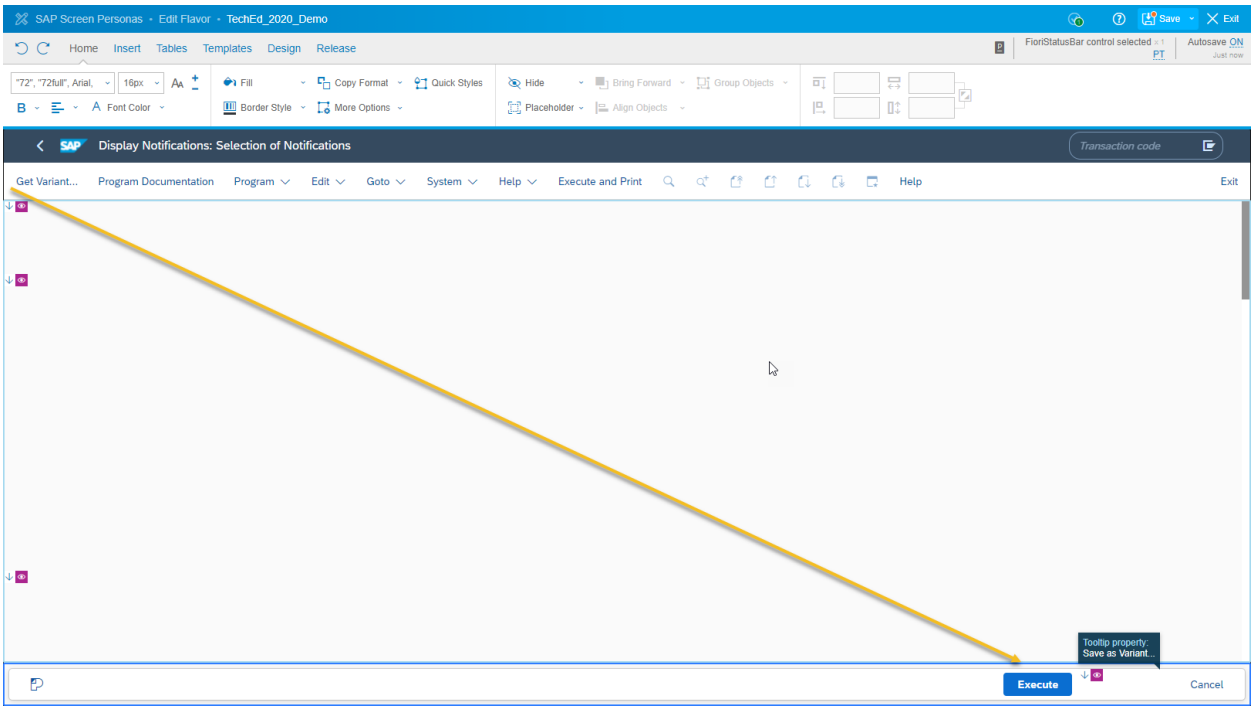
We only need the first ('From') selection for Equipment and Functional Location, and the notification date range.



Let's hide all other selections. Since they are grouped in boxes, we only need to hide those container group boxes. Multi-select them by the box label and hit the 'Hide' button in the toolbar's 'Home' tab. Don't forget to scroll down and hide every remaining group box and selection field. The selection screen is quite tall.

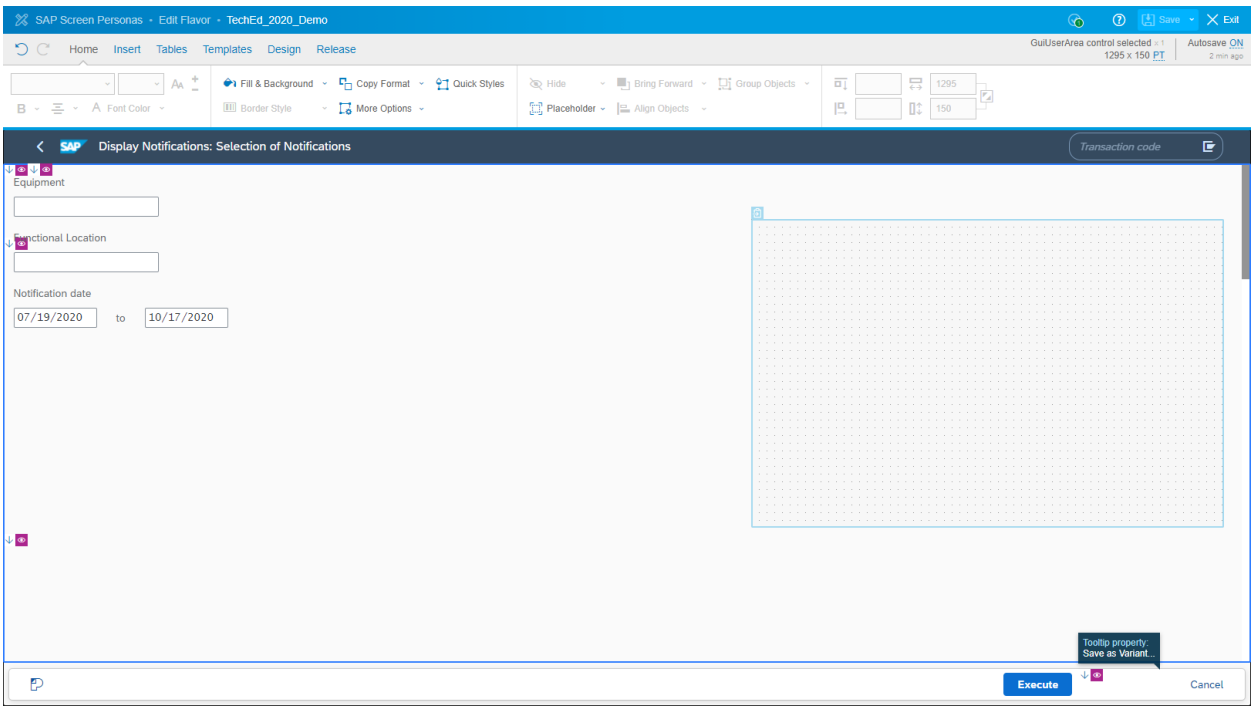

SAP Screen Personas • Edit Flavor • TechEd_2020_Demo
Save
Exit
Home
Insert
Tables
Templates
Design
Release
Group controls selected
Autosave ON
Fill & Background
Copy Format
Quick Styles
Hide
Bring Forward
Group Objects
Border Style
More Options
Placeholder
Align Objects
Display Notifications: Selection of Notifications
Transaction code
Execute
Get Variant...
Program Documentation
Program
Edit
Goto
System
Help
Execute and Print
Help
Exit
Notification status
Outstanding
Postponed
In process
Completed
Sel.profil:
Addr.
Equipment:
Functional Location:
Notification date:
07/19/2020
to:
Notification selection
Notification:
Notification Type:
Material:
Serial Number:
Addit. device data:
Order:
Partners:
Cls.
General Data/Administrative Data
Description:
Created by:
Created on:
Save as Variant...
Cancel

Now, drag the ‘Execute’ button of the application toolbar to the footer, next to ‘Save as Variant’, then hide ‘Save as Variant’.

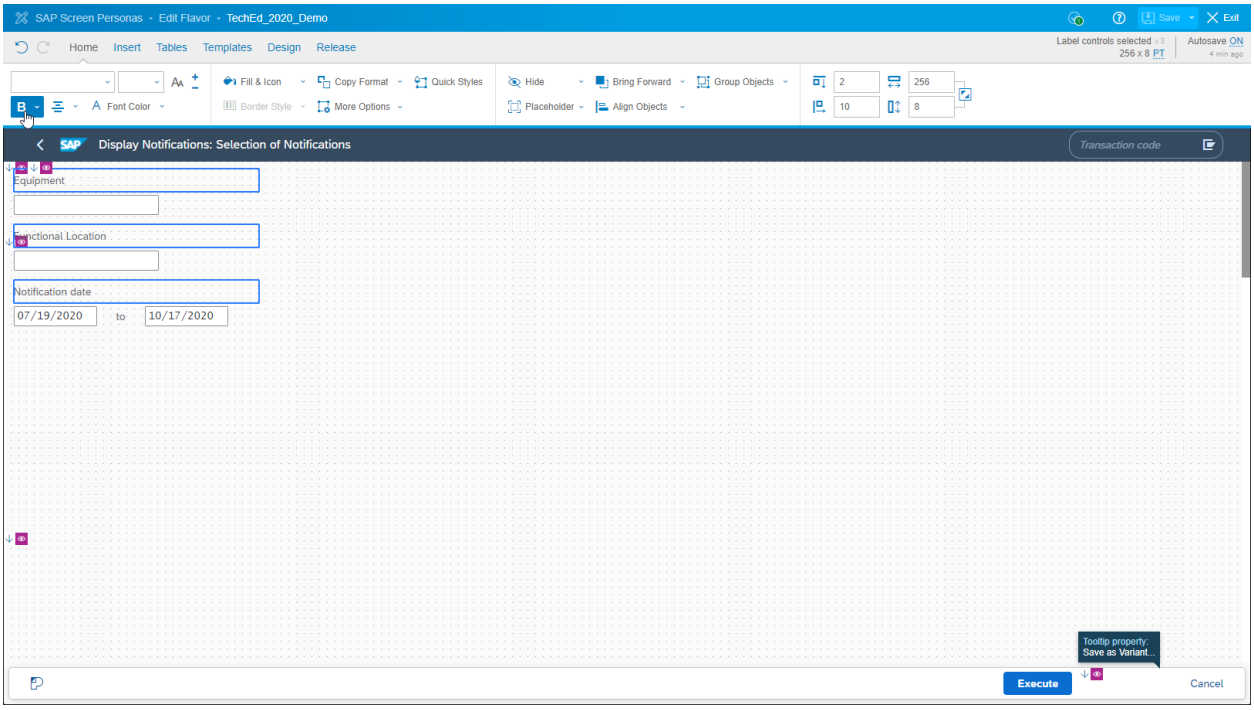


Hide the application’s menu bar (at the top) as well, since we will not need other buttons from it.

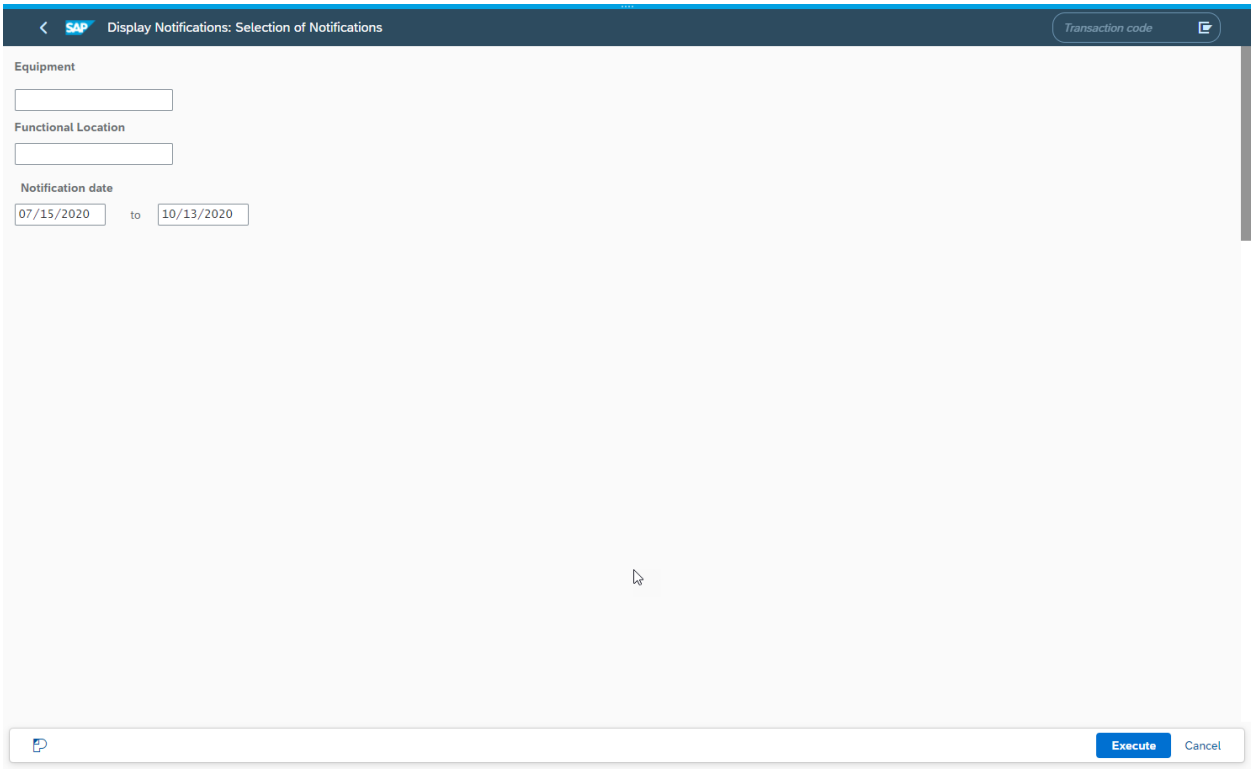
Select the ‘Design’ tab in the Personas toolbar and bring back the ‘Parking Lot’ where the stored selection fields are. Start moving them to their final position, so the screen looks like this:



Close the ‘Parking Lot’. Select the three labels, make their font bold.



Now the selection screen is done:



The only additional issue is that if the user changes the selection values on this screen, the current ‘Back’ button script in transaction IW53 will not bring back the previous result list, since other than the

notification type set in our flavor, all other selections will use the default values. So, we need to save the selection screen values and as part of the 'back' script, we need to reapply them. Using the session store is the simple solution for this.

This means that the 'Execute' button function must be enhanced with storing the selections. Here is the new 'Execute' script:

```
// Store selection values
session.utils.put('equipmentNr', session.findById("wnd[0]/usr/ctxtEQUNR-LOW").text);
session.utils.put('funcLoc', session.findById("wnd[0]/usr/ctxtSTRNO-LOW").text);
session.utils.put('dateFrom', session.findById("wnd[0]/usr/ctxtDATUV").text);
session.utils.put('dateTo', session.findById("wnd[0]/usr/ctxtDATUB").text);

// Press 'Execute'
session.findById("wnd[0]/tbar[1]/btn[8]").press();

// Suppress standard 'Execute' button
return true;
```

Add this to the 'Execute' button's *onClick* event, like you did this earlier with 'back' in transaction IW53.

Finally, this is the snippet you need to add to the 'back' script in IW53. It should go after starting transaction IW29, but before clicking 'Execute':

```
// Re-populate selection screen fields
session.findById("wnd[0]/usr/ctxtEQUNR-LOW").text = session.utils.get('equipmentNr');
session.findById("wnd[0]/usr/ctxtSTRNO-LOW").text = session.utils.get('funcLoc');
session.findById("wnd[0]/usr/ctxtDATUV").text = session.utils.get('dateFrom');
session.findById("wnd[0]/usr/ctxtDATUB").text = session.utils.get('dateTo');
```