1. Mocking the WebApp

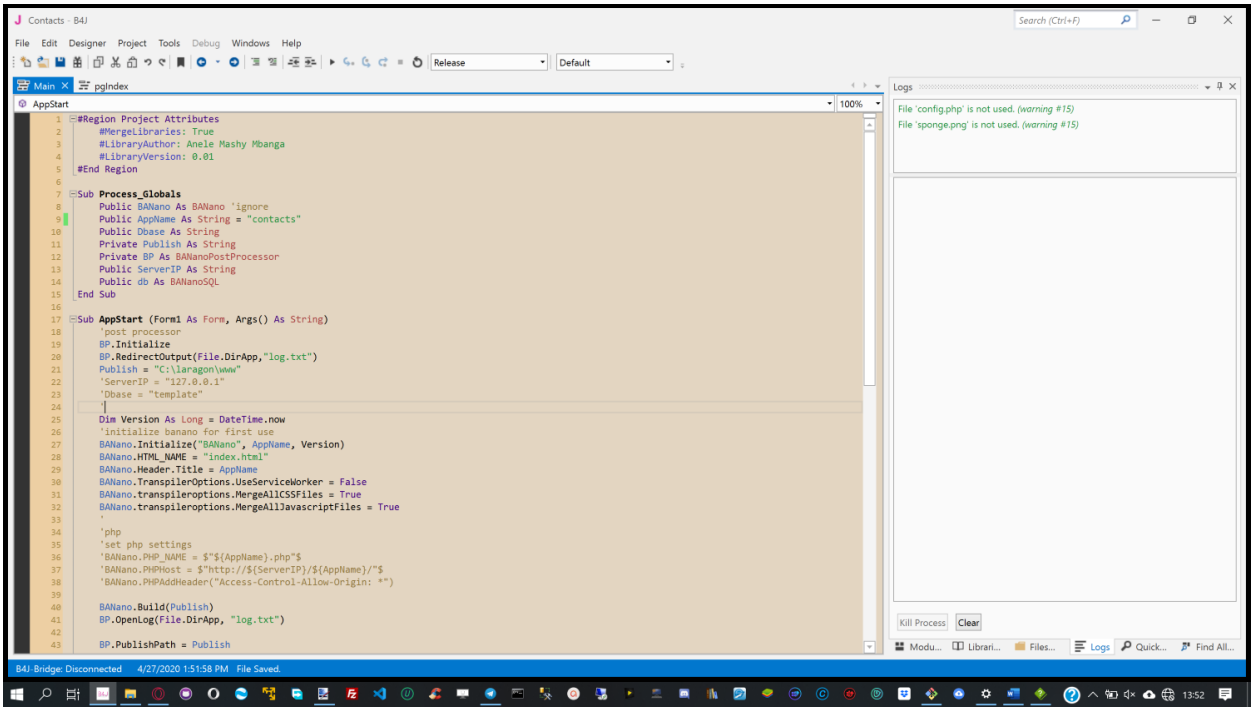
1. Copy the “5. Template” folder to a new folder
2. Rename this new folder to “Contacts”
3. Open the newly renamed folder and rename the “Template.b4j” file to “Contacts.b4j”
4. Delete the Template.b4j.meta file

You should have this…

Name	Date modified	Type	Size
Files	2020/04/24 16:45	File folder	
Objects	2020/04/27 13:48	File folder	
Contacts.b4j	2020/04/27 13:48	B4J Source Code	3 KB
Contacts.b4j.meta	2020/04/27 13:48	META File	1 KB
pgIndex.bas	2020/04/24 20:42	BAS File	1 KB

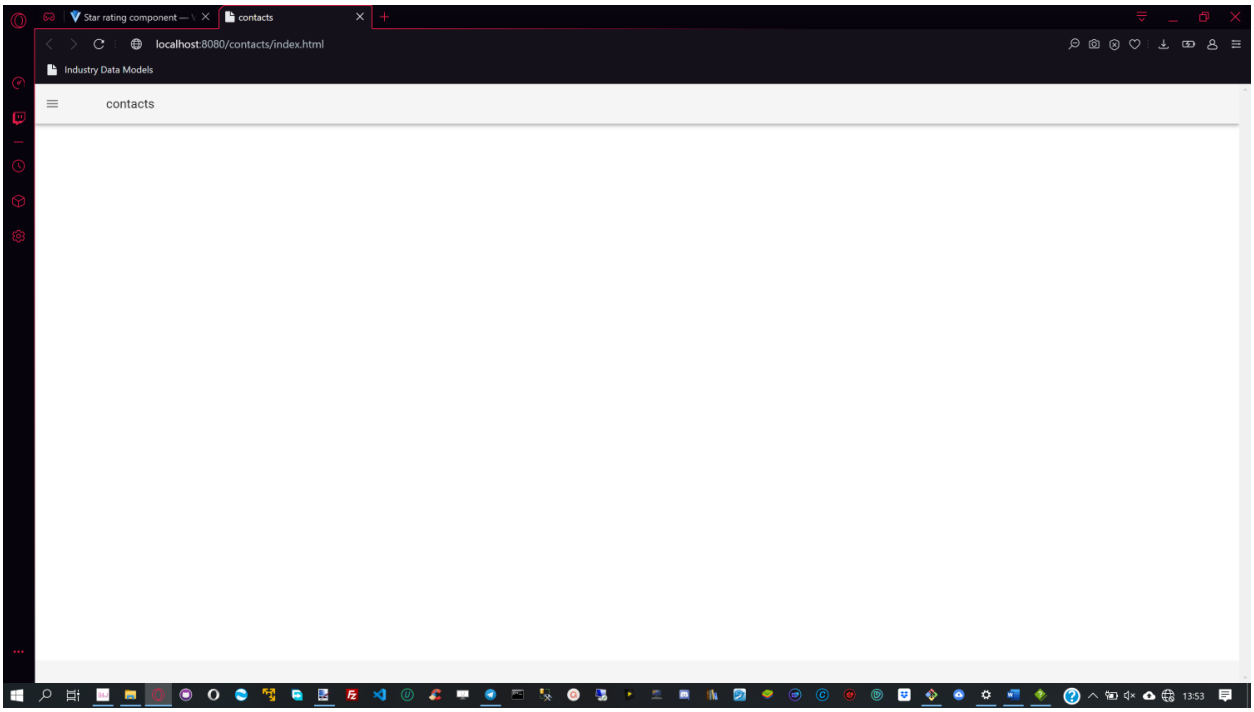
5. Double click the “Contacts.b4j” file – this should open up the project

Updating the “Main” module



1. Update line 3 to reflect your name. This information wont be used anyway as we will not compile this to a library but are just producing an app.
2. Update line 9 by replacing “template” with the name of the app. We will call the app “contacts”. This could be any name you want your app to be called. Please note that the app name links to the compilation folder for the app. This will be **c:\laragon\www\contacts**

3. We are publishing our app to laragon, if you use xampp or any other server, update the publish folder to be what you use.
4. The starting page for our app will be index.html has indicated in line 28
5. You can change the title of the page to be what you need. For now its linked to the appname, meaning that it will be “contacts”, also in lowercase.
6. For this app we are not using service workers, so you can leave line 30 as is. Turning this to true will ensure that our app is available offline.
7. We are also not using PHP for our app, so leave lines 36 – 38 commented.
8. Our laragon is configured to run in port 8080, so we leave line 50 as is, otherwise change this to be the port your webserver is running on.
9. Compile the app, if all goes well, you should see this.



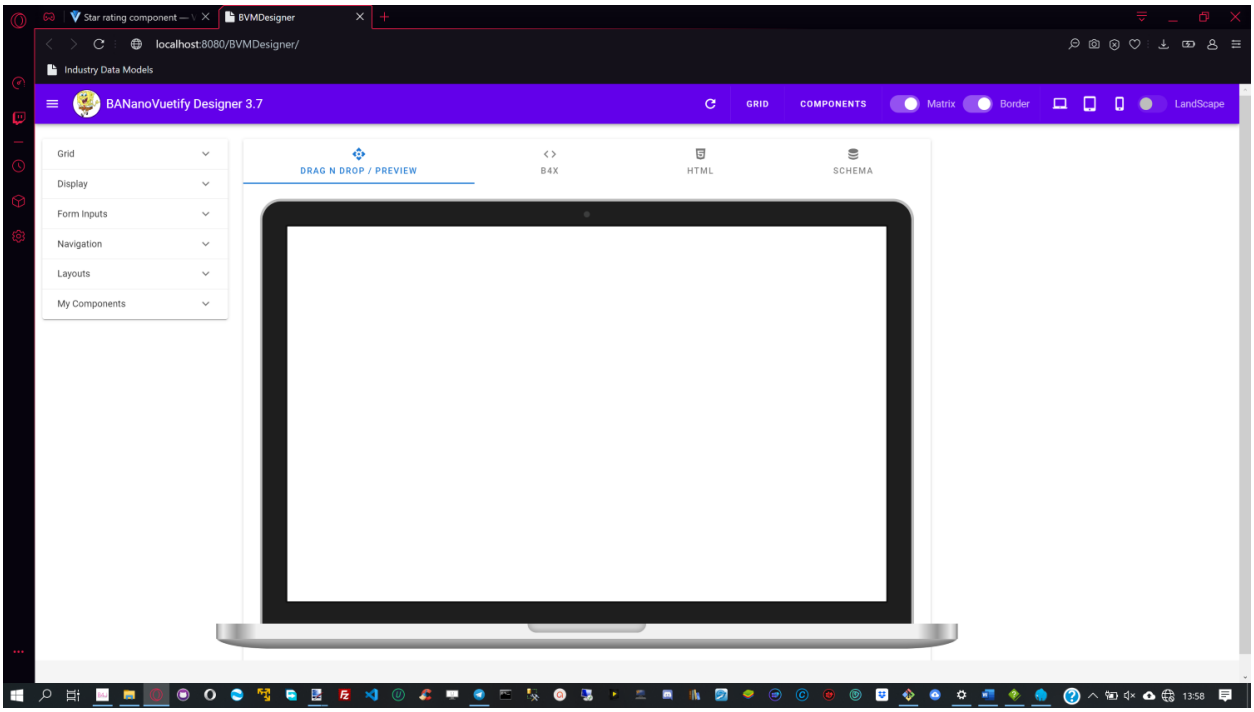
Congratulations, you just ran your first BVM App.

Let’s create a data entry screen. We want to keep a record of our contacts. We need to record the following information for our contacts:

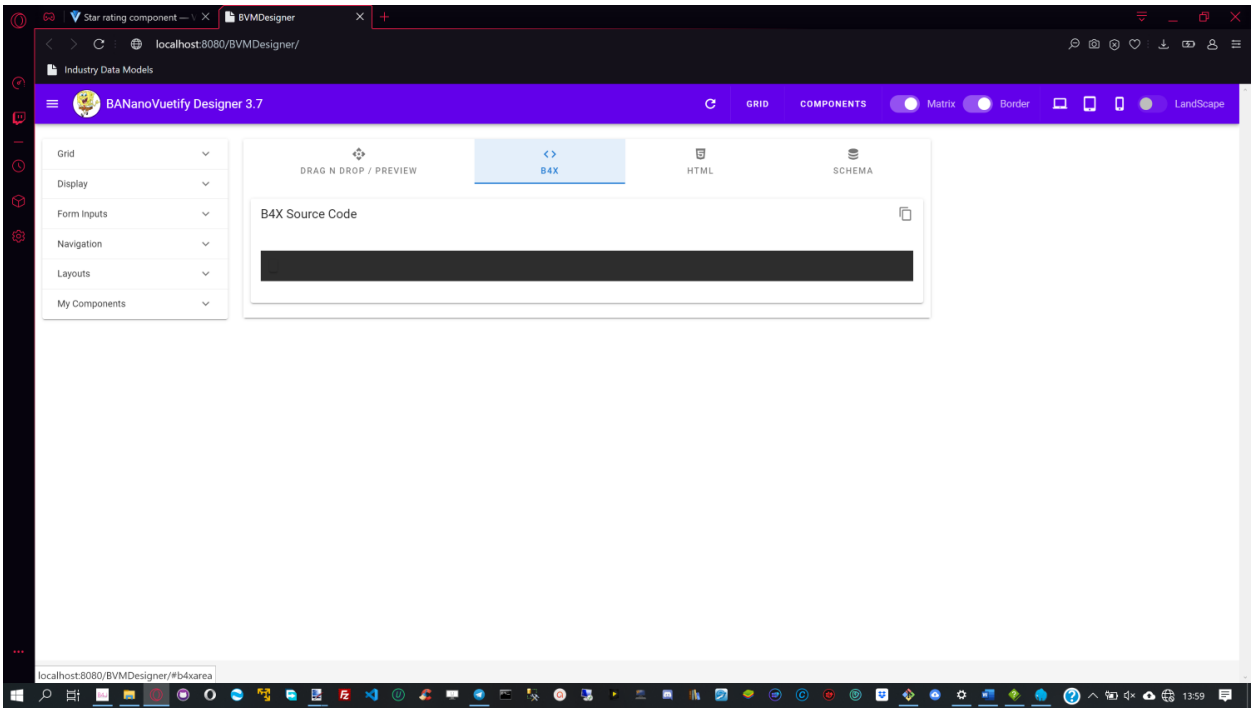
1. First Name (text)
2. Last Name (text)
3. Mobile Number (tel)
4. Email Address (email)
5. Date of Birth (date picker)
6. Gender (radio)
7. Receive Notifications (switch)

8. City (auto complete)

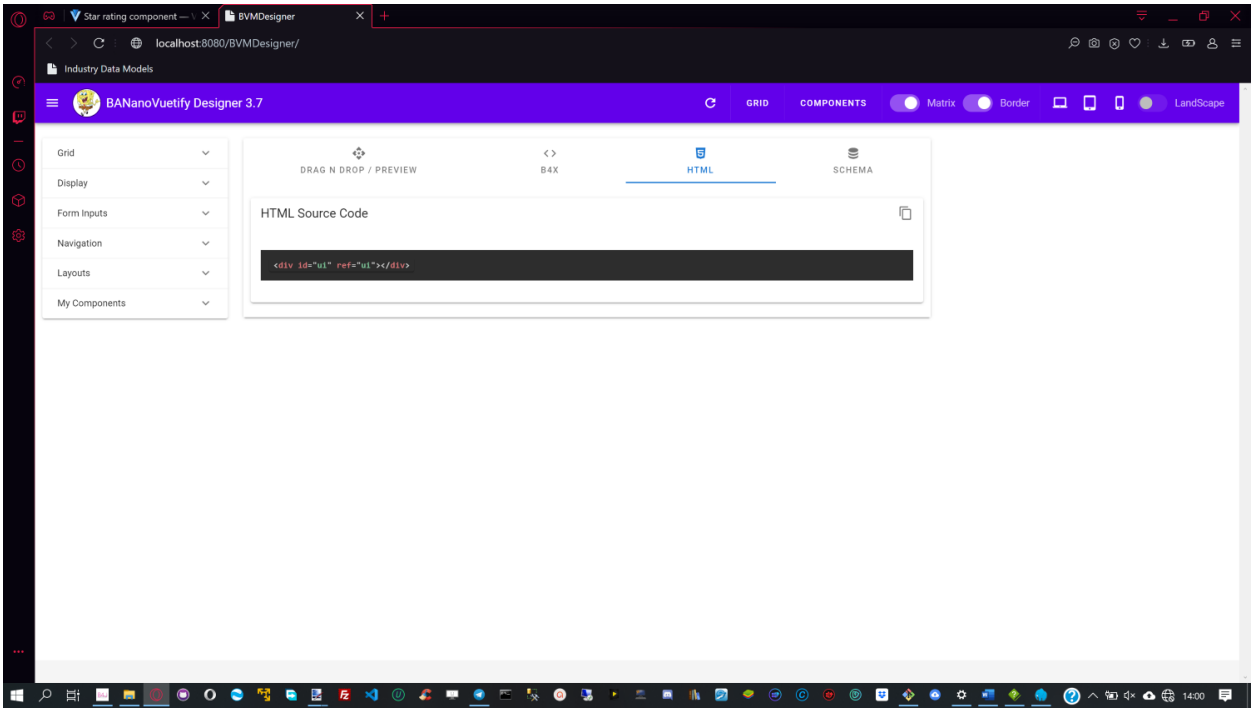
We will use the designer to do this. Let's activate the BVMDesigner. As we are starting a new project, ensure that the designer is blank. In the toolbar, click Components > Clear. If you had used components from the Grid section in the toolbar, also execute Grid > Clear



Check and ensure that the B4X code section is also not having any code.



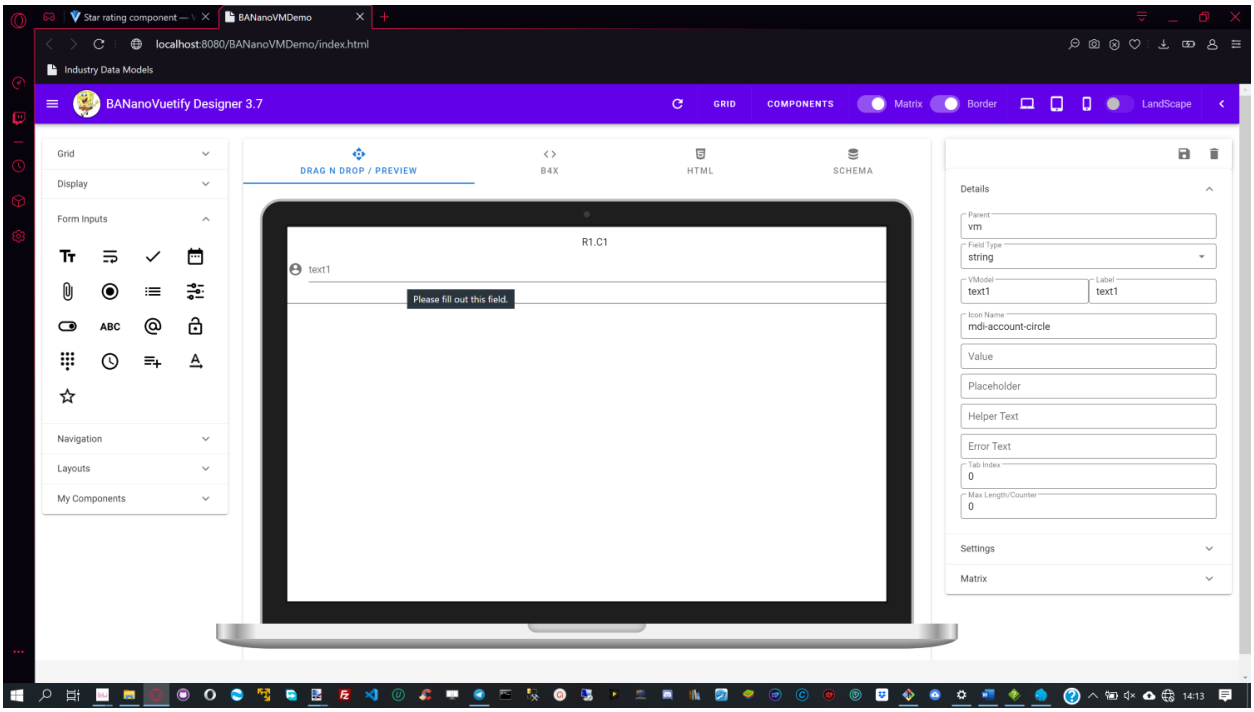
The HTML code section should also reflect this.



If you are not in the “Drag n Drop / Preview” tab, ensure you select it to activate it.

Adding the First Name

1. In the toolbox, activate the “Form Inputs” panel. Drag and drop a “Text field” to the stage. You should see this.

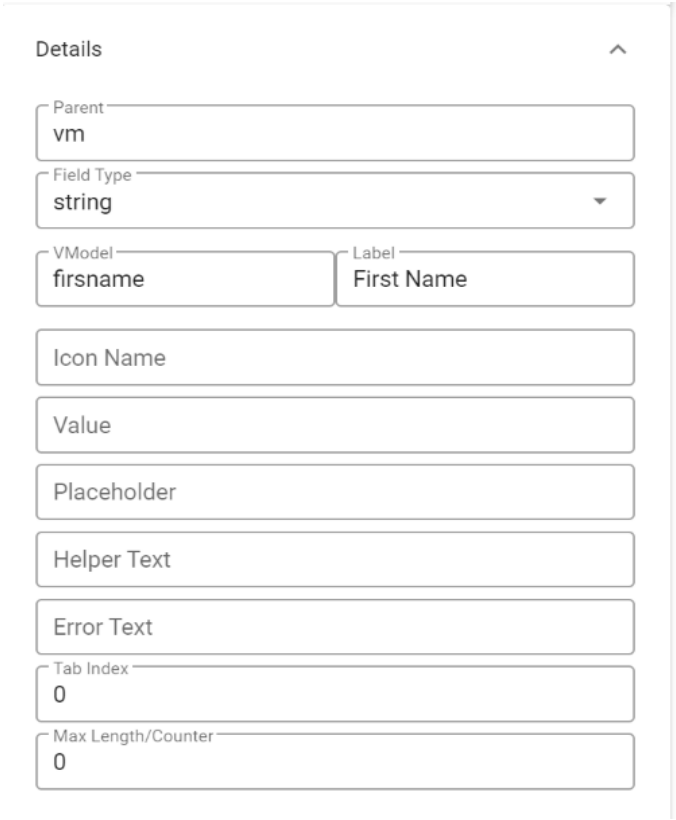


The name of this text field is “text1”, we need to change it to reflect the firstname. Expand the “My Components” panel in the toolbar. Click “text1” on the list, this activates the property bag for the text field.

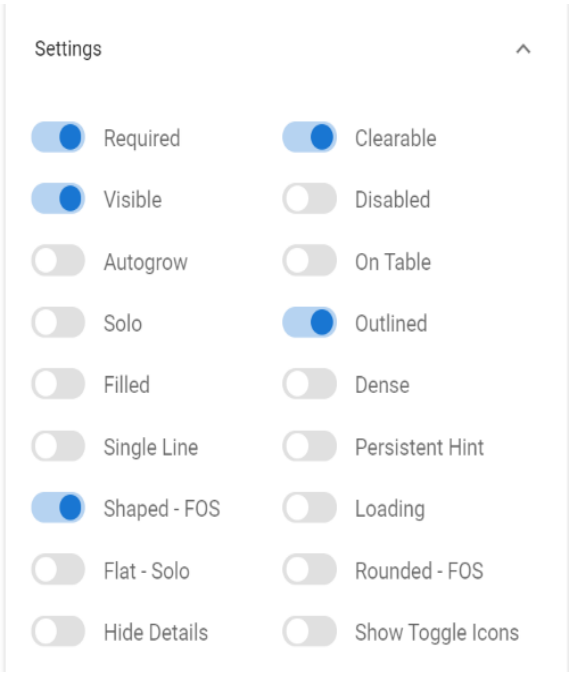
We are adding the first name directly on the page, so we will leave the “**parent**” as “**vm**” in the property bag. If we were adding this component to a container, we would specify the name of that container in the “parent” field. To navigate from one element to another in the property bag, use the TAB key in your keyboard.

Change the “**vmodel**” to be **firstname**. This is essence is the unique identifier for this control. This is like the field name that will be used to get / set the value of this control. Press the TAB key and then update the “label” to “First Name” and press TAB key.

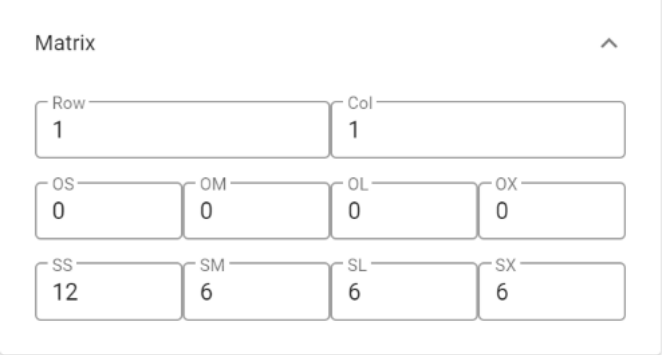
Delete the content in the “**Icon Name**” as we do not need an prepend icon for the firstname. The final “Details” section of the property bag should be:



Activate the “Settings” panel of the property bag. Turn on “**Outlined**” and also turn on “**Shaped – FOS**”. This should now look like:

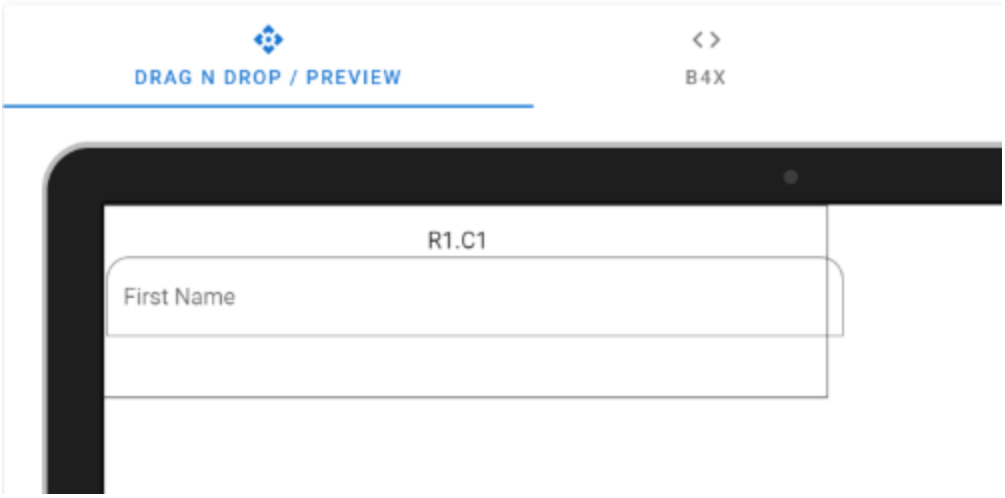


Activate the "Matrix", the matrix helps you define the location and size of the component ROW/COL co-ordinates. We want the first name to take half of the span in the row. For this we need to change the sizes. Let's update the matrix values to reflect the content below: Change **SM, SL and SX** from 12 to **6**.



This means on cellphones, the firstname will take the complete row whilst on medium, large and xtra large devices, half of the row space will be used. The first name is located on R1C1 i.e row 1 and column 1. We are now complete with the design of our first name text field where an end user will enter content.

The content of the stage should now look like this:

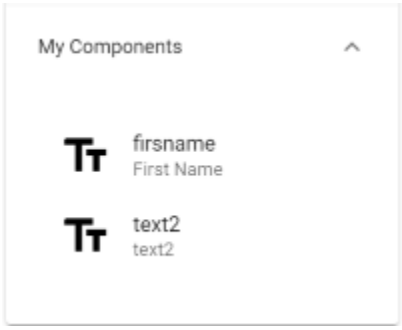


Because we have the Matrix and Border in the toolbar active, we can toggle these on the toolbar to hide them. Let's do that. Update your toolbar switches to be off.

Now, the border and matrix props don't appear in the stage.

Adding the Last Name

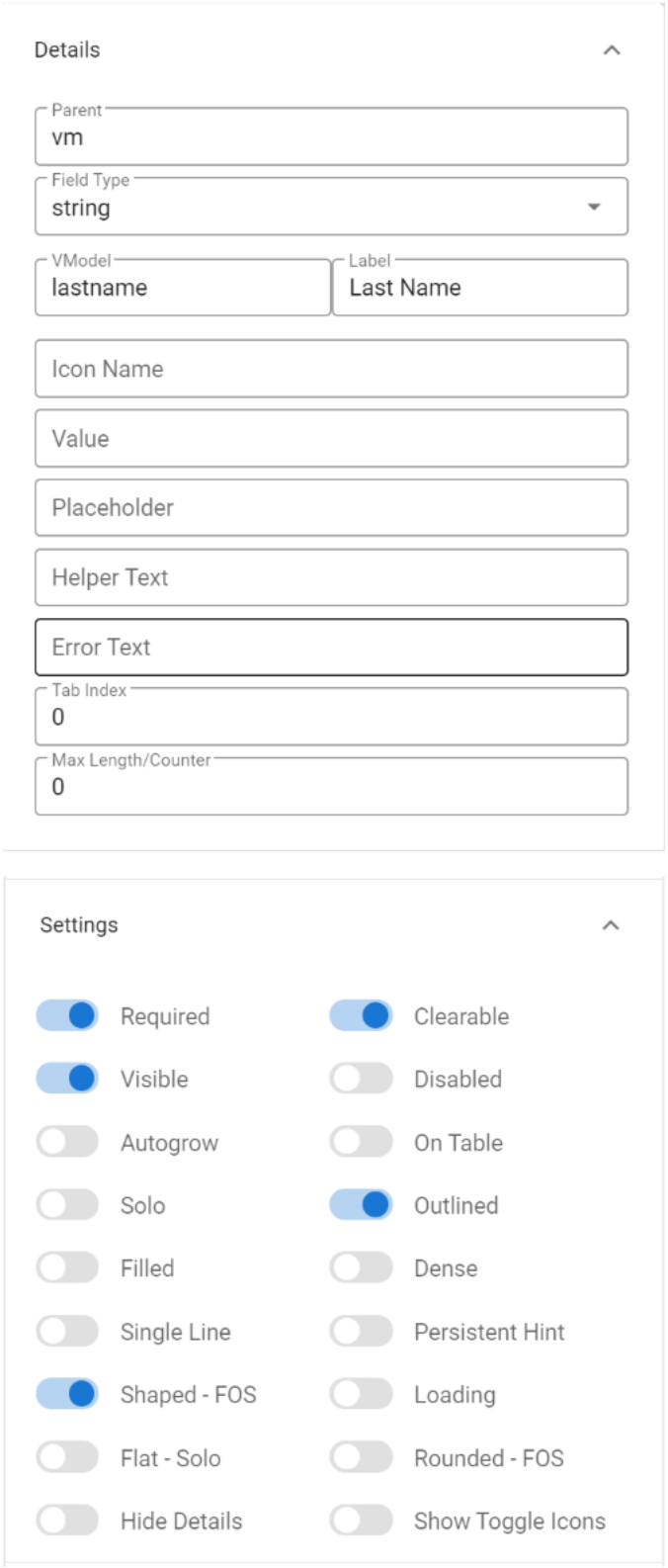
From the Toolbox, Form Inputs panel, drag n drop another text field to the stage. You will notice now that when you activate the "My Components" section, 2 items are listed as indicated below:

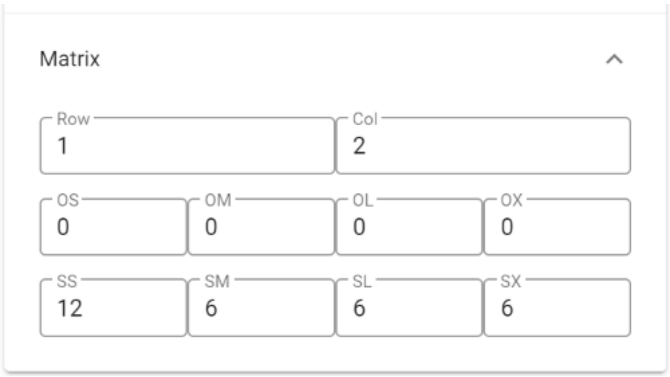


Click the text2 item in the list, this activates the property bag for text 2. We need to change this to lastname. Note that text2 is now placed on R2C1 as R1C1 already has a component. We however need the last name to appear on the same row as first name on the right.

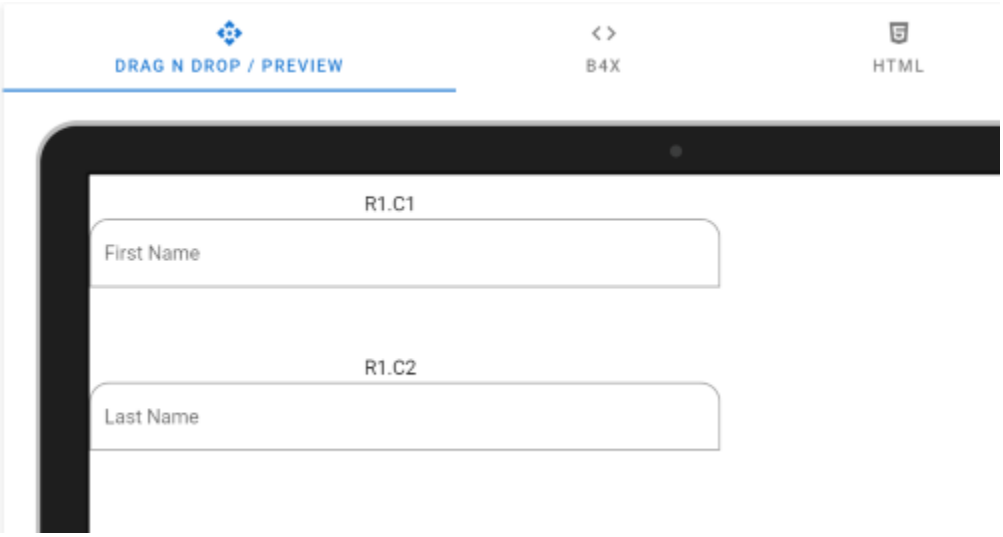
R1C1 is spanning 6 spaces. We can then make last name, be located on R1C2 and also spanning 6 spaces. Let's update the property bag for text2 to reflect the lastname properties.

Now you should have this by updating **text2 to lastname**


Details
Parent
vm
Field Type
string
VModel
lastname
Label
Last Name
Icon Name
Value
Placeholder
Helper Text
Error Text
Tab Index
0
Max Length/Counter
0
Settings
Required
Clearable
Visible
Disabled
Autogrow
On Table
Solo
Outlined
Filled
Dense
Single Line
Persistent Hint
Shaped - FOS
Loading
Flat - Solo
Rounded - FOS
Hide Details
Show Toggle Icons



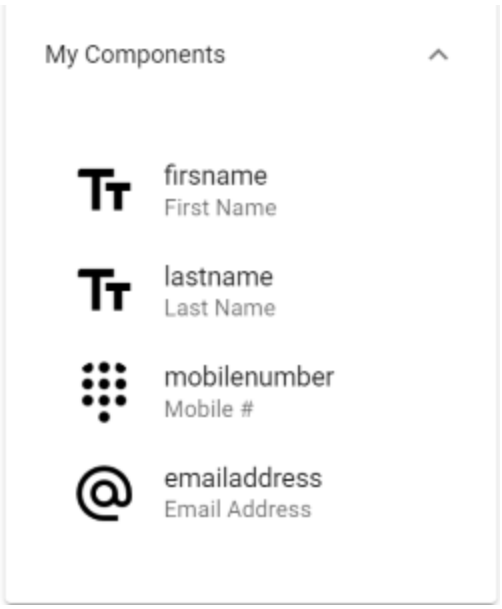
If you activate the Matrix now on the toolbar, you will notice how the elements "**will**" be placed.



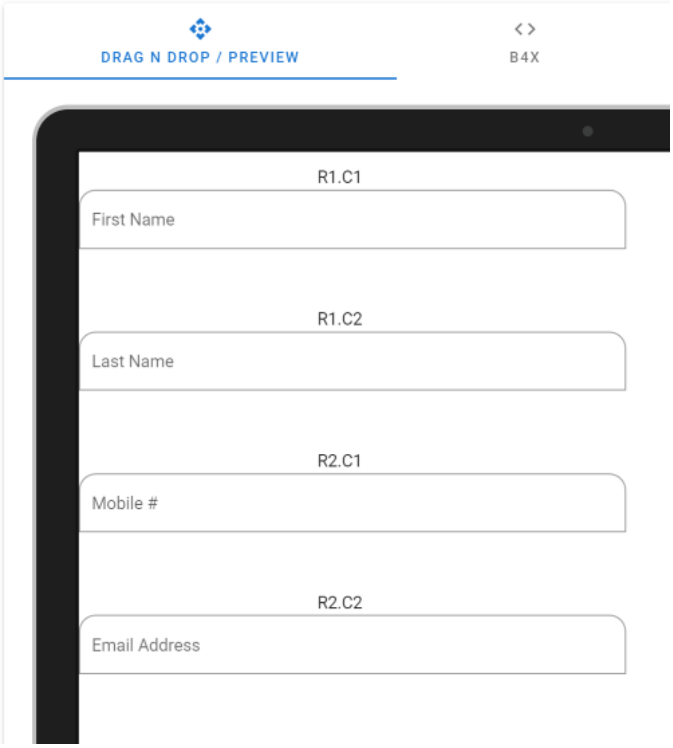
So far we are on the right track. Now lets do the same process below. Let's add a mobile number (My Inputs > Telephone) and Email Address (My Inputs > Email). We will place these on R2C1 and R2C2 respectively, both spanning 6 spaces each.

Adding the Mobile Number and Email Address.

When complete, you should have something like this under "My Components"



The stage should now look like this.



We can also preview the B4X source code that we will compile. This should now show:

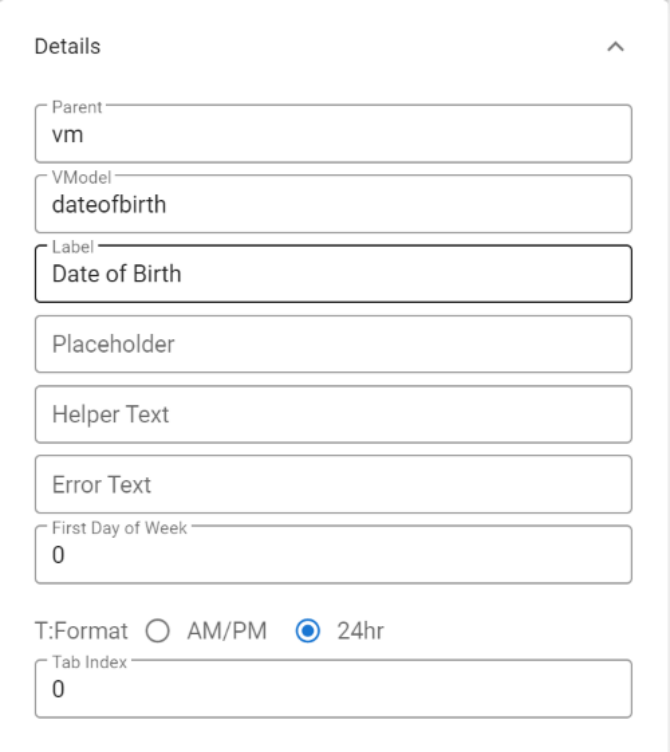
B4X Source Code

```
 Dim txtfirsname As VMTextField = vm.NewTextField(Me, true, "txtfirsname", "firsname", "First Name", "", true, "", 0, "", "", 0)
txtfirsname.SetOutlined(true)
txtfirsname.SetShaped(true)
txtfirsname.SetClearable(true)
vm.Container.AddControl(txtfirsname.textfield, txtfirsname.tostring, 1, 1, 0, 0, 0, 0, 12, 6, 6, 6)

Dim txtlastname As VMTextField = vm.NewTextField(Me, true, "txtlastname", "lastname", "Last Name", "", true, "", 0, "", "", 0)
txtlastname.SetOutlined(true)
txtlastname.SetShaped(true)
txtlastname.SetClearable(true)
vm.Container.AddControl(txtlastname.textfield, txtlastname.tostring, 1, 2, 0, 0, 0, 0, 12, 6, 6, 6)

Dim telmobilenumber As VMTextField = vm.NewTel(Me, true, "telmobilenumber", "mobilenumber", "Mobile #", "", true, "", "", "", 0)
telmobilenumber.SetOutlined(true)
telmobilenumber.SetShaped(true)
telmobilenumber.SetClearable(true)
vm.Container.AddControl(telmobilenumber.textfield, telmobilenumber.tostring, 2, 1, 0, 0, 0, 0, 12, 6, 6, 6)

Dim emailemailaddress As VMTextField = vm.NewEmail(Me, true, "emailemailaddress", "emailaddress", "Email Address", "", true, "", "",
emailemailaddress.SetOutlined(true)
emailemailaddress.SetShaped(true)
emailemailaddress.SetClearable(true)
vm.Container.AddControl(emailemailaddress.textfield, emailemailaddress.tostring, 2, 2, 0, 0, 0, 0, 12, 6, 6, 6)
```

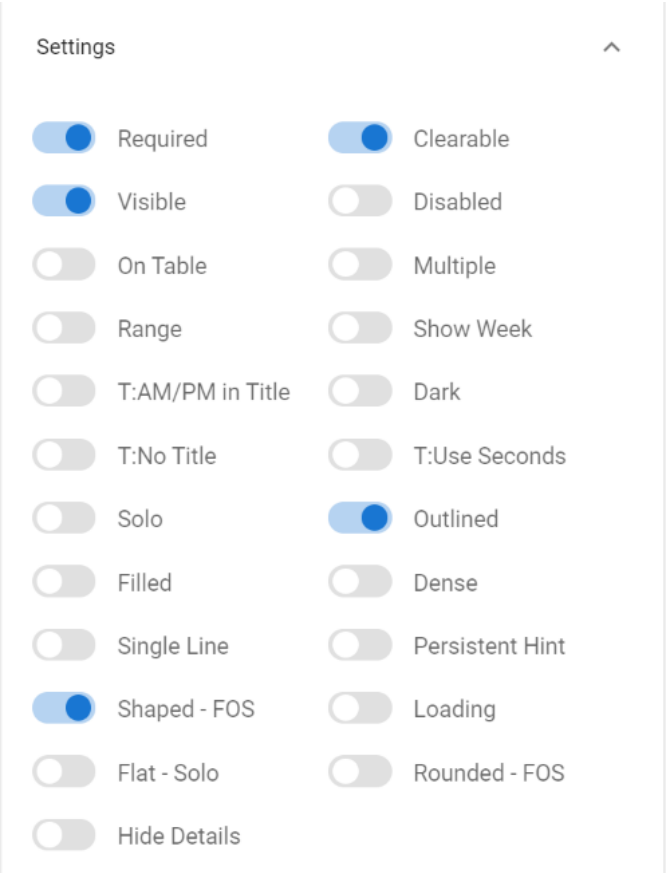
However, we are not finished, we need to add a date picker and a radio for users to select a gender. We want the Date of Birth and Gender to be on the same row, i.e R3. We are adding components in sequence of placement, thus its important to follow the matrix placement structure in sequence, Row 1, Row 2, Row 3 etc.

Adding the Date of Birth

Activate the “Form Inputs” and drag n drop a “Date Picker” to the stage. Under “My components” select the added date picker, this activates the property bag. Drag n Drop a “Date Picker” to the stage and update it to reflect these changes.

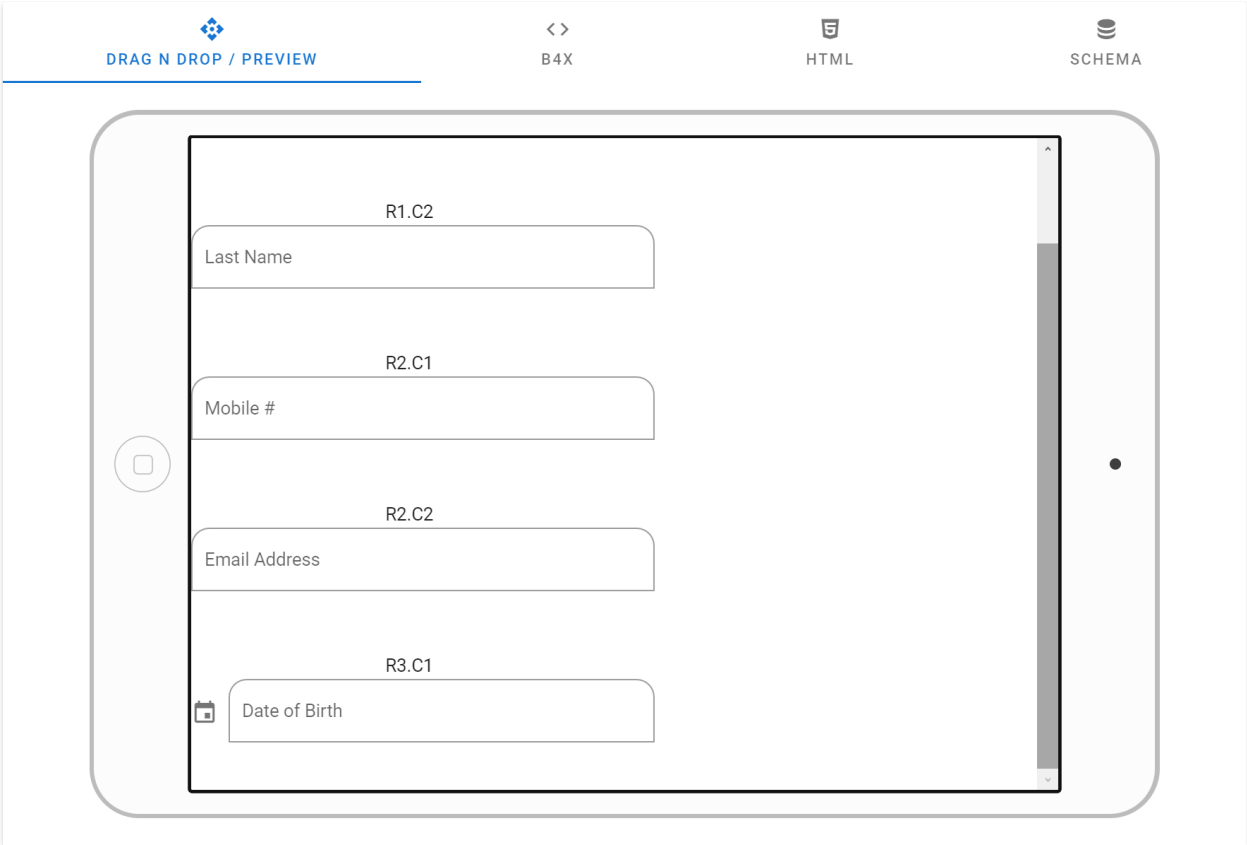


We also need it outlined and shaped. So update the “Settings”



Also update the Matrix for the date picker so that it's taking half of Row 3.

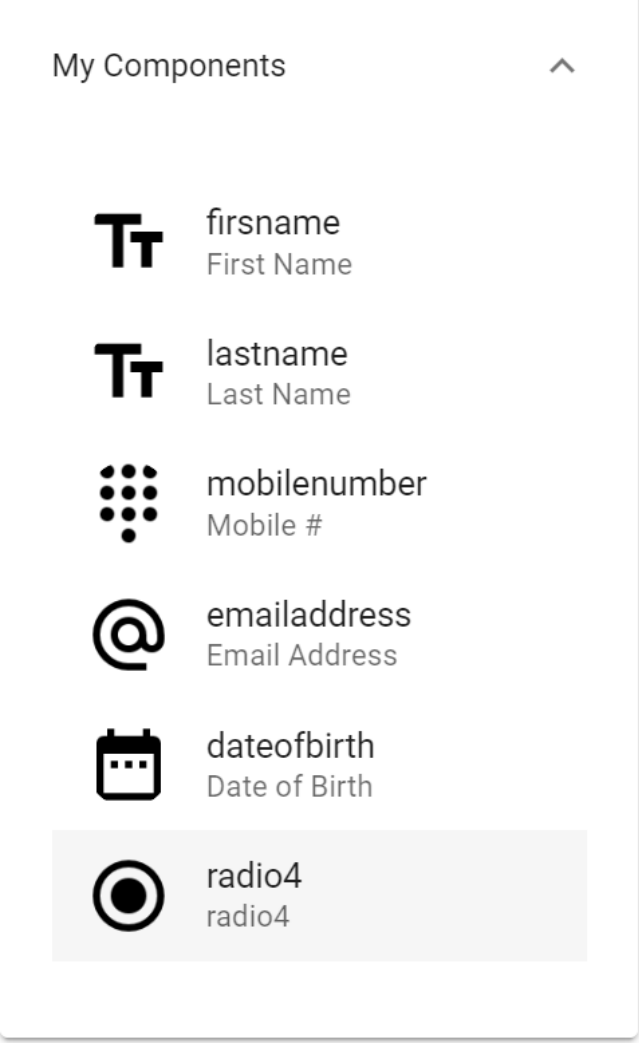
When done, our stage should now look like below when we change the device to be an iPad in LandScape mode.



Adding the Gender

We need the gender to also be in the third row. We will place this on R3C2 also taking 6 spans.

Drag n Drop the "Radio Group" component from the toolbox. Under "My Components", select the radio control and activate the property bag.



Our radio for gender will source the items to display from a list of key value pairs i.e. list.map. The keys for the map will be M and F and the values will be Male and Female. The keys will be stored in a "field" name called "id" and the values stored in a "field" name called "text". So we need something like this:

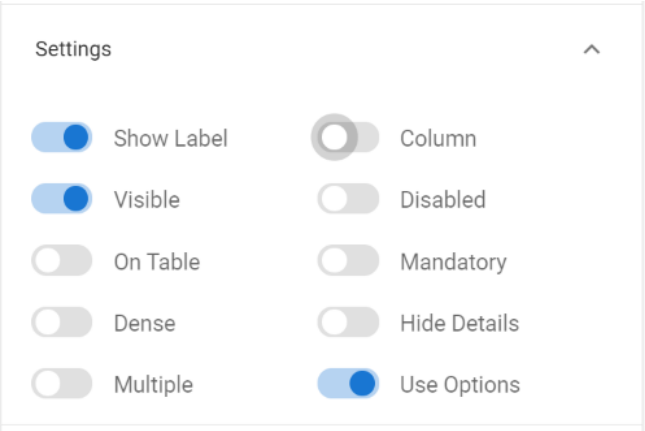
```
Dim gendertypes As List

gendertypes.Initialize

gendertypes.Add(CreateMap("id":"M", "text":"Male"))

gendertypes.Add(CreateMap("id":"F","text":"Female"))
```

This will be the **vue state** we will use to source our gender list. When you select "Male", the component will return "M" and when you select "Female" the component will return "F". Let's create this with the designer. Update the property bag for the gender to be:

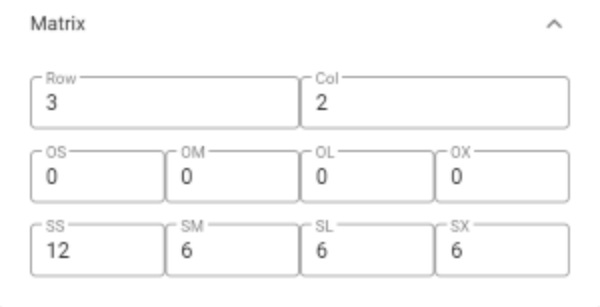
Internally, the code will build this list.map record as a state object called “gendertypes” using.

```
Vm.SetData(“gendertypes”, gendertypes)
```

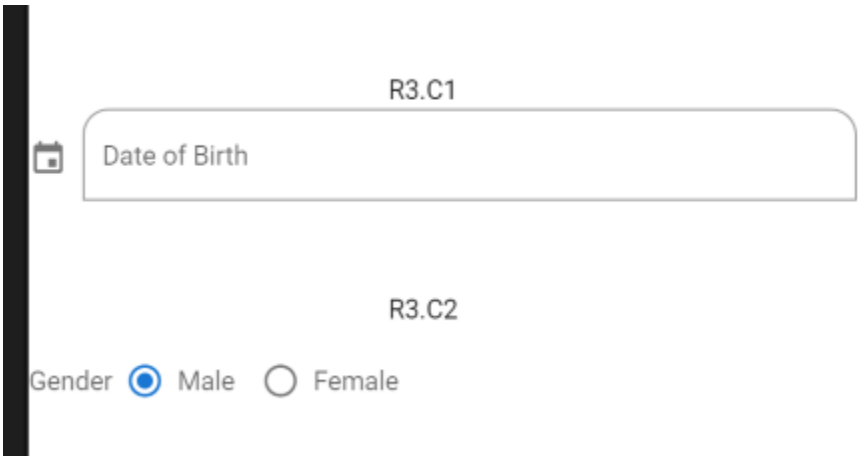
We need the items in the radio to appear in horizontal fashion. Let’s update the settings and turn “**column**” off. The “**Use Options**” is indicating that we are using items defined by the designer key, values pairs. Should we source items from a database, we would turn this option off.



We then update the matrix for the radio group so that its on the same row as the date of birth.



Row 3 should now look like this in the designer:



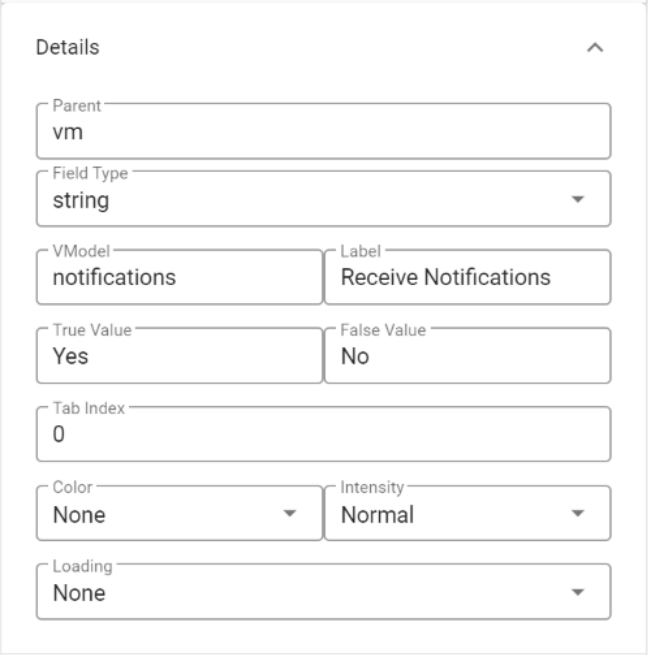
Congratulations, you are almost done if you can get to this stage.

Now we need to add a switch so that the contact can receive notification. We also need to add an auto complete so that we can select which city the person lives in.

Adding Receive Notifications

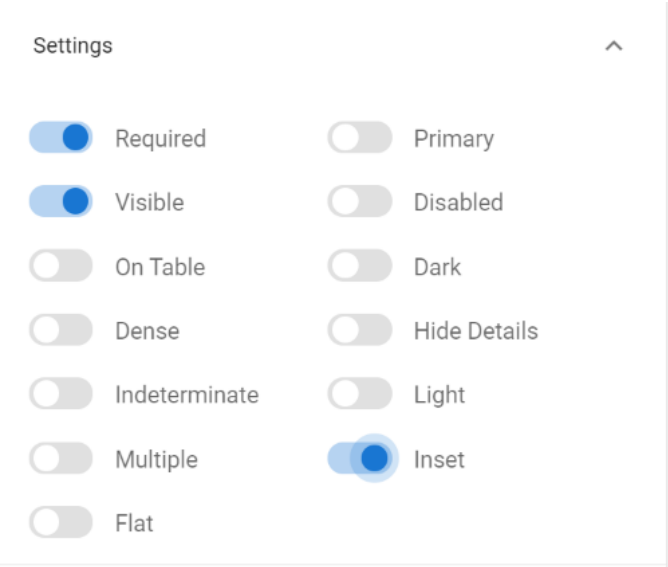
Let's add a switch on R4C1 spanning 6 spaces and then we will add a city also on R4C2 spanning 6 spaces.

Under "Form Inputs" in the toolbox, drag n drop a Switch component to the stage. Select the switch under "My Components" to activate the property bag. Update the property bag for the radio to reflect these changes.

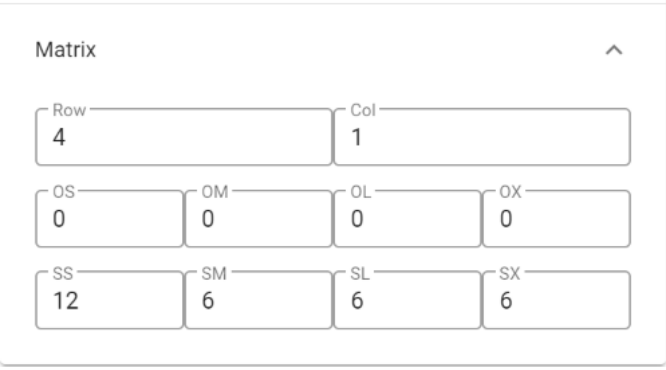


NB: From above, when the switch is turned off, the value to return will be “No” and when switched on, the return value will be “Yes”. This is the default settings.

We want this switch to be like the ones used in our property bag. Let’s change a setting to make it like that. Update the settings for the switch to be like this:



By turning Inset to true, the switch is automatically updated on the stage. We want the switch to take half of the row space. Lets update the SM, SL and SX to reflect 6 spans.



We are done with the switch. Now let's add a city. We will use an auto-complete for this so that a person can select the city. We will use our own defined list for the cities, also based on a map just like the radio group above.

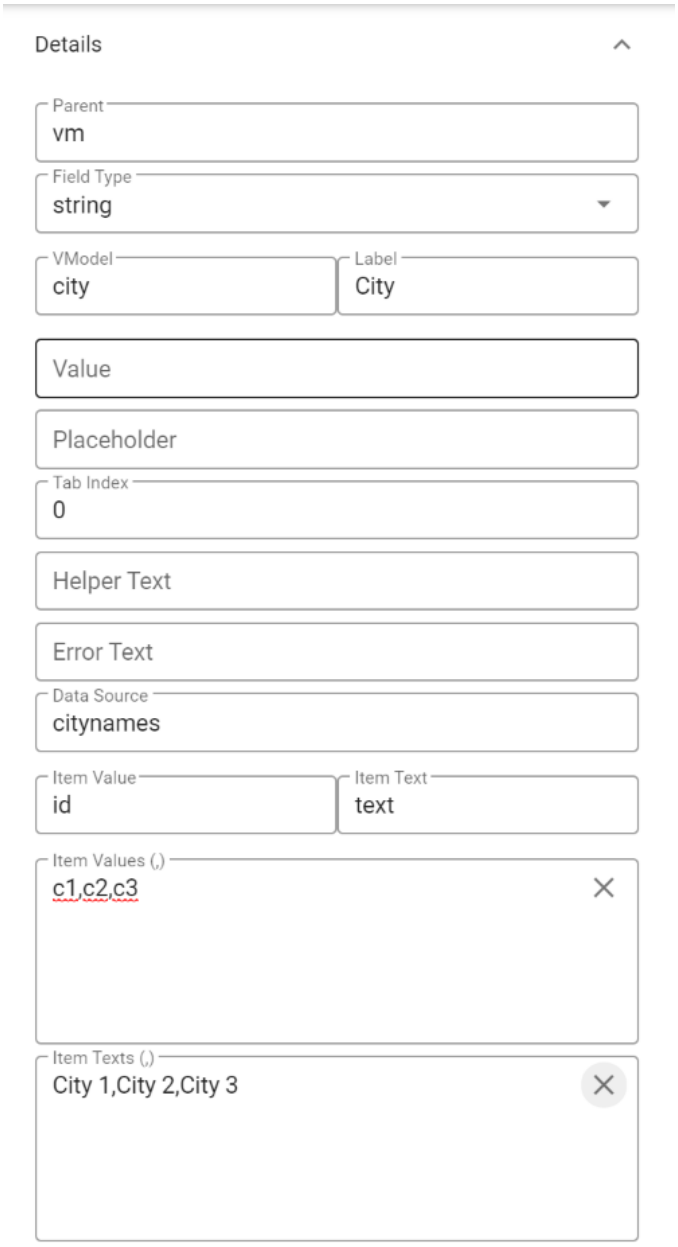
Adding a City

Drag n Drop and Auto Complete component from "Form Inputs" to the stage. Select the auto complete component from the list under "My Components" to activate the property bag.

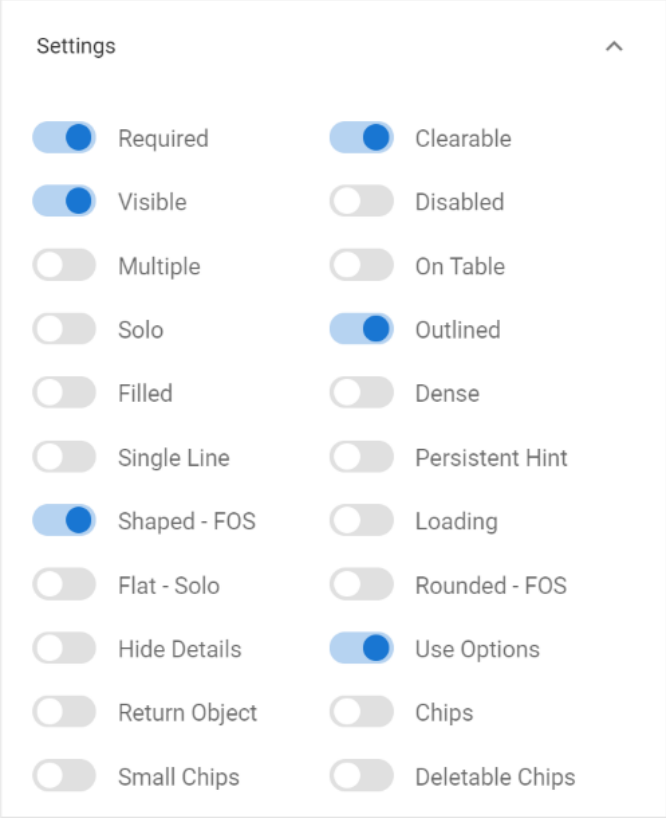
Just like the radio, we will define our own source for the data using key value pairs.

```
Dim citynames As List

Citynames.Initialize

Citynames.add(CreateMap("id":"c1", "text":"City 1"))

Citynames.add(CreateMap("id":"c2", "text":"City 2"))

Citynames.add(CreateMap("id":"c3", "text":"City 3"))
```

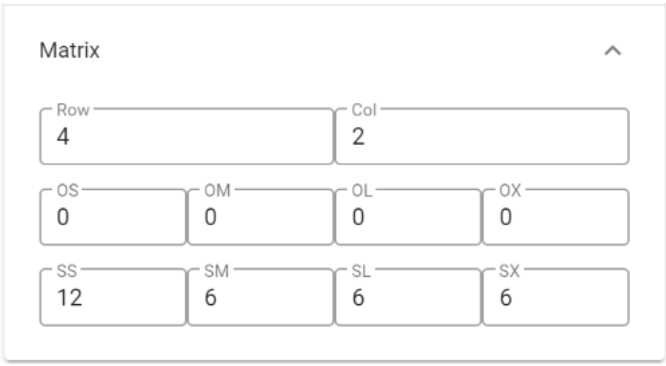
So also, when **City 1** is selected, the returned key will be **c1**



Update the “Settings” for the auto-complete component so that its shaped and outlined.



Update the Matrix so that the auto-complete sits on R4C2 and takes 6 spaces.

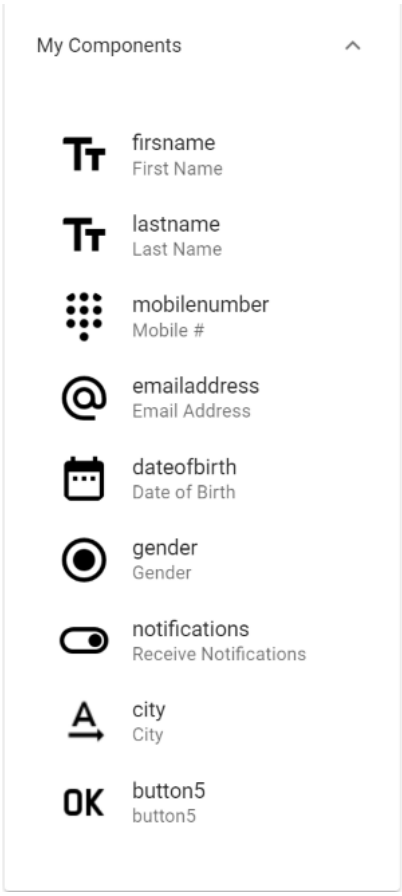


Adding a button

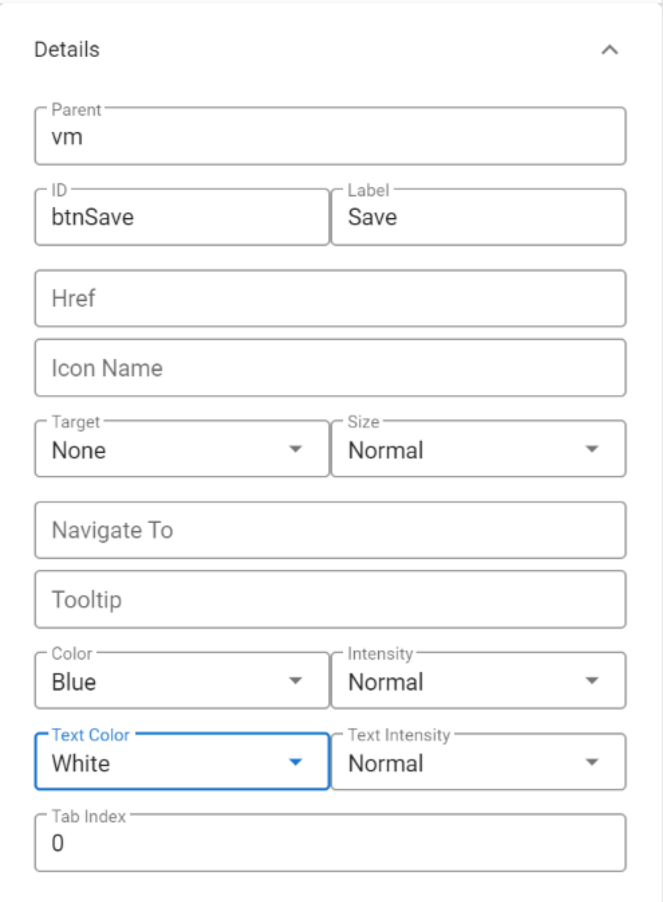
Now let's add a button that will get the contents from each component on the page and then saves it as a map. This map can be used to save records to a database table.

In the toolbox activate the "Navigation" panel and drag and drop a button to the stage. This will be placed on the center of the RC, don't worry about that. This is due to the iMac being centered on the div.

Under "My Components", you should be seeing, all the components.



Click on the button to activate the property bag and then update the property bag to reflect the needed changes. The button property bag should reflect these changes.

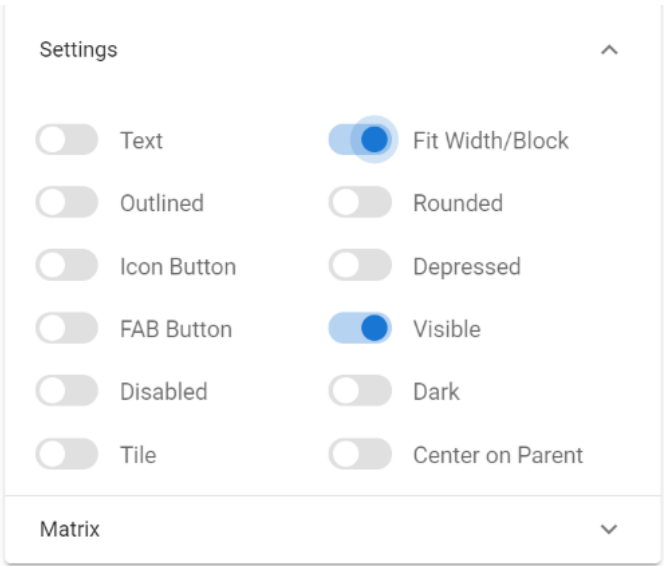


The unique identifier for the button is btnSave. This will expect a click event called btnSave_click.

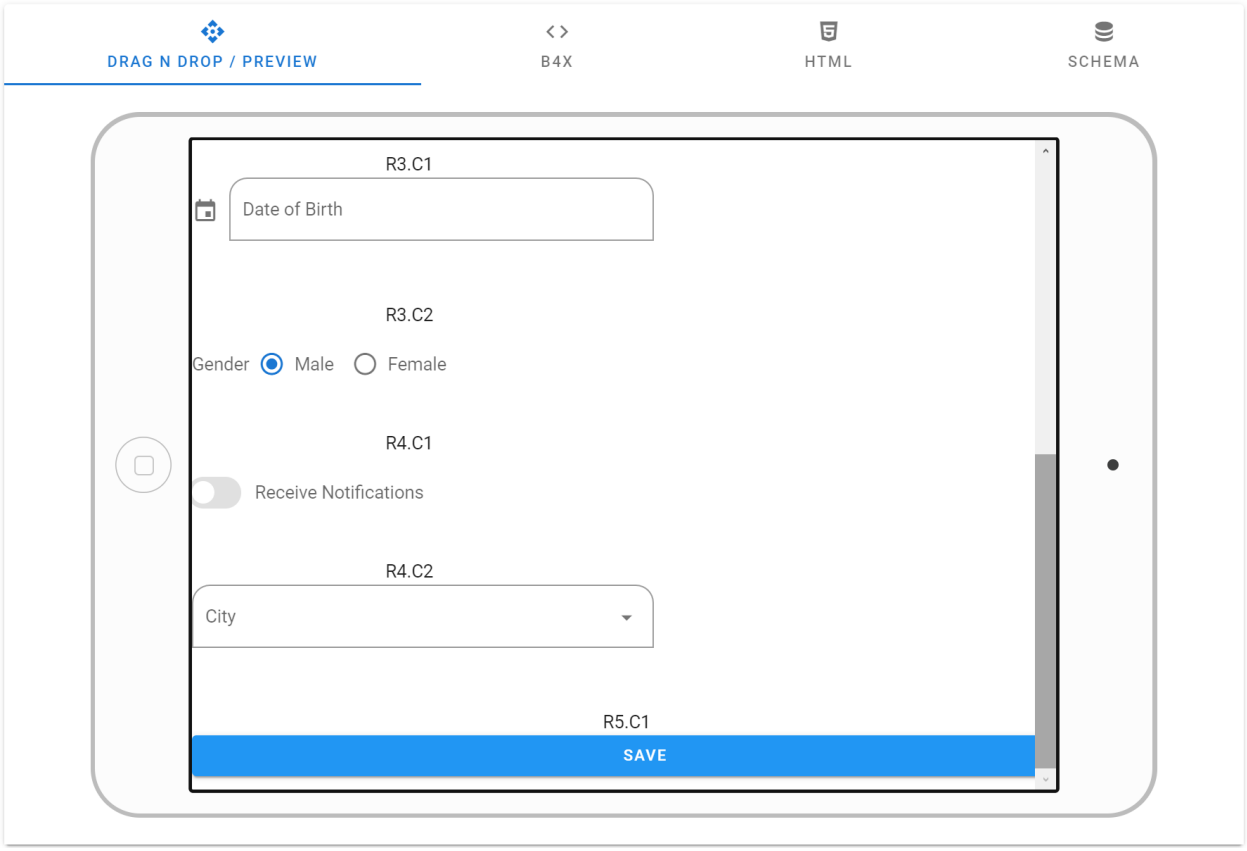
So in our code, we need to define an event like this:

```
Sub btnSave_click(e As BANanoEvent)

End Sub
```

The button will have a blue back-ground and white text. We want the button to fit the whole RC space. Let's update the Settings to reflect these changes.

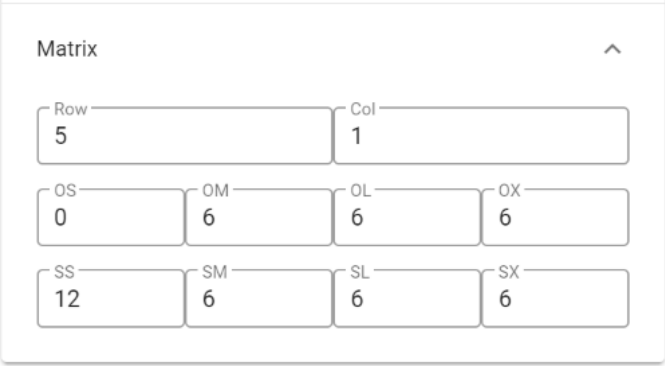


This is what should appear in the stage.



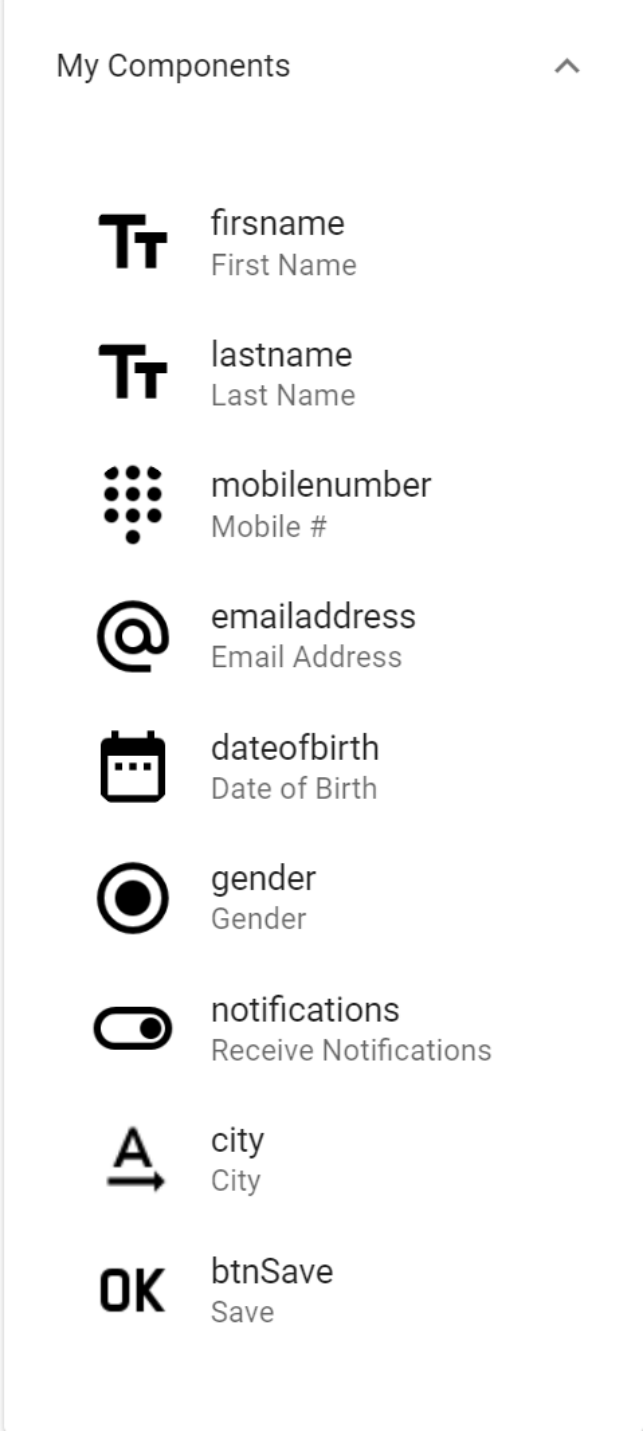
Lets try and nudge the button to the right. We can do this by updating the offsets. We want the button to be on the right part of the row. Half of the page is 6 spaces, lets then nudge the button 6 spaces.

R5C1 is currently taking/spanning 12 spaces. So to nudge the button to the right, the button should be within the 12 spans. We nudge the button 6 spaces to the right and then make it to take 6 spaces. This equals a span of 12.



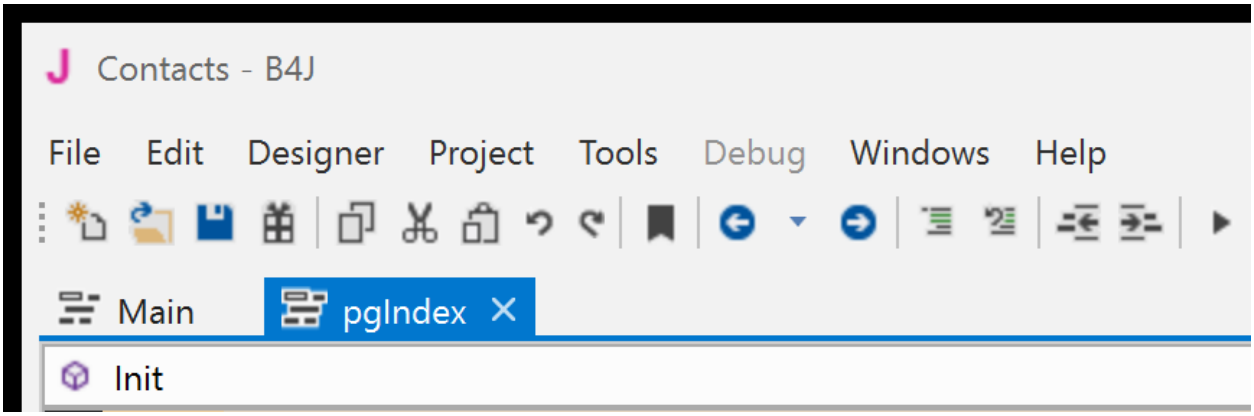
On small devices, we don't want any nudging taking place as we want the size to be 12. This will create this effect on the button.

If all goes well, all your components should be listed under "My Components" in the following manner:



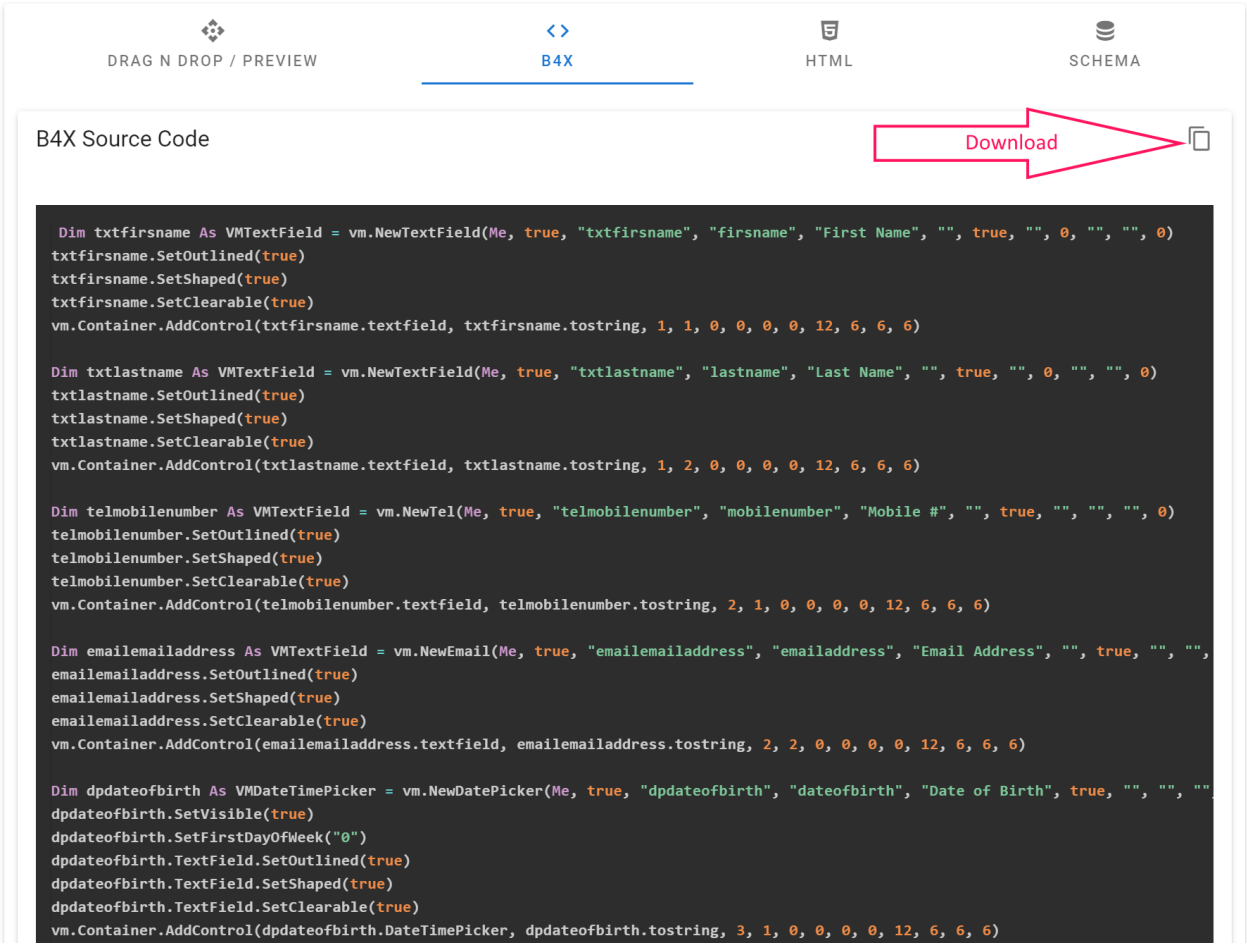
2. **Compiling the Web App**

Now we need to copy the generated B4X code from our designer to our "Contacts" project. The entry point for our WebApp is the **pgIndex** code module. As we are not going to create multiple pages for this app, we will just use the generates source code in this module.



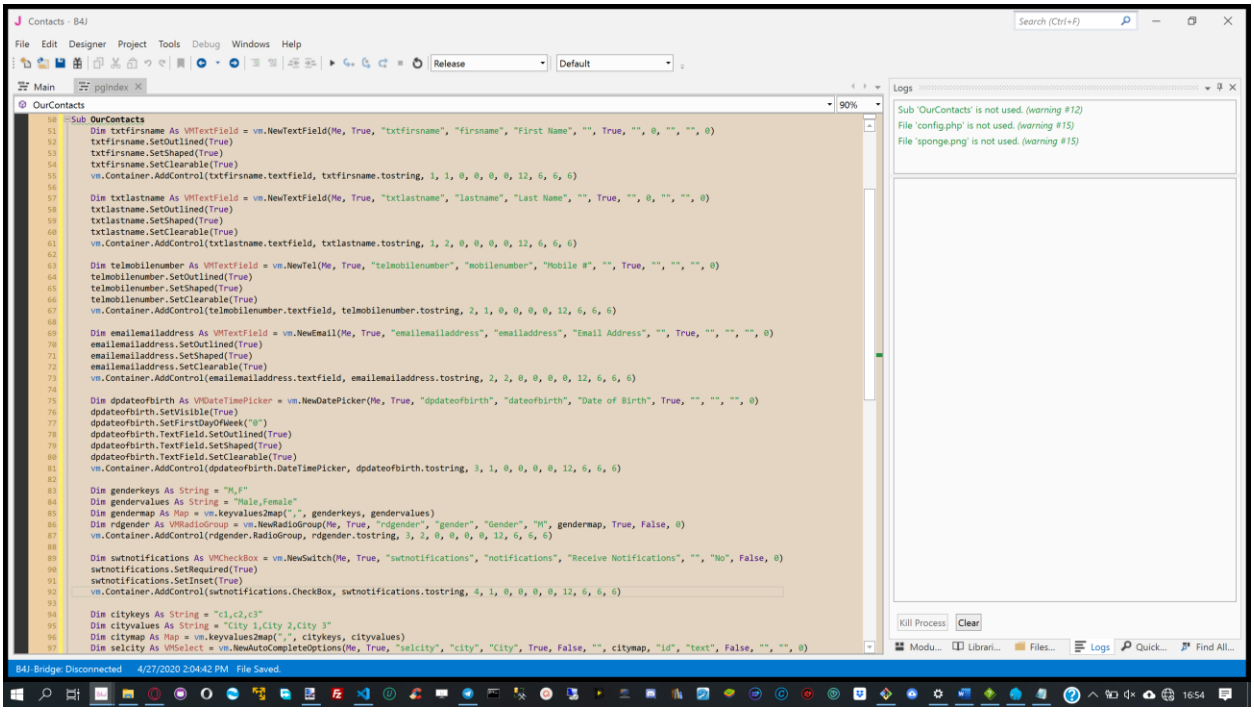
Lets create a subroutine in this module and call it. **OurContacts** and then paste code inside it.

In the designer, activate the "**B4X**" tab and download the source code by clicking the download button. Open the downloaded file to be able to copy the source code.




```
Dim txtfirsname As VMTextField = vm.NewTextField(Me, true, "txtfirsname", "firsname", "First Name", "", true, "", 0, "", "", 0)
txtfirsname.SetOutlined(true)
txtfirsname.SetShaped(true)
txtfirsname.SetClearable(true)
vm.Container.AddControl(txtfirsname.textfield, txtfirsname.tostring, 1, 1, 0, 0, 0, 0, 12, 6, 6, 6)

Dim txtlastname As VMTextField = vm.NewTextField(Me, true, "txtlastname", "lastname", "Last Name", "", true, "", 0, "", "", 0)
txtlastname.SetOutlined(true)
txtlastname.SetShaped(true)
txtlastname.SetClearable(true)
vm.Container.AddControl(txtlastname.textfield, txtlastname.tostring, 1, 2, 0, 0, 0, 0, 12, 6, 6, 6)

Dim telmobilenumber As VMTextField = vm.NewTel(Me, true, "telmobilenumber", "mobilenumber", "Mobile #", "", true, "", "", "", 0)
telmobilenumber.SetOutlined(true)
telmobilenumber.SetShaped(true)
telmobilenumber.SetClearable(true)
vm.Container.AddControl(telmobilenumber.textfield, telmobilenumber.tostring, 2, 1, 0, 0, 0, 0, 12, 6, 6, 6)

Dim emailemailaddress As VMTextField = vm.NewEmail(Me, true, "emailemailaddress", "emailaddress", "Email Address", "", true, "", "",
emailemailaddress.SetOutlined(true)
emailemailaddress.SetShaped(true)
emailemailaddress.SetClearable(true)
vm.Container.AddControl(emailemailaddress.textfield, emailemailaddress.tostring, 2, 2, 0, 0, 0, 0, 12, 6, 6, 6)

Dim dpdateofbirth As VMDateTimePicker = vm.NewDatePicker(Me, true, "dpdateofbirth", "dateofbirth", "Date of Birth", true, "", "", ""
dpdateofbirth.SetVisible(true)
dpdateofbirth.SetFirstDayOfWeek("0")
dpdateofbirth.TextField.SetOutlined(true)
dpdateofbirth.TextField.SetShaped(true)
dpdateofbirth.TextField.SetClearable(true)
vm.Container.AddControl(dpdateofbirth.DateTimePicker, dpdateofbirth.tostring, 3, 1, 0, 0, 0, 0, 12, 6, 6, 6)
```

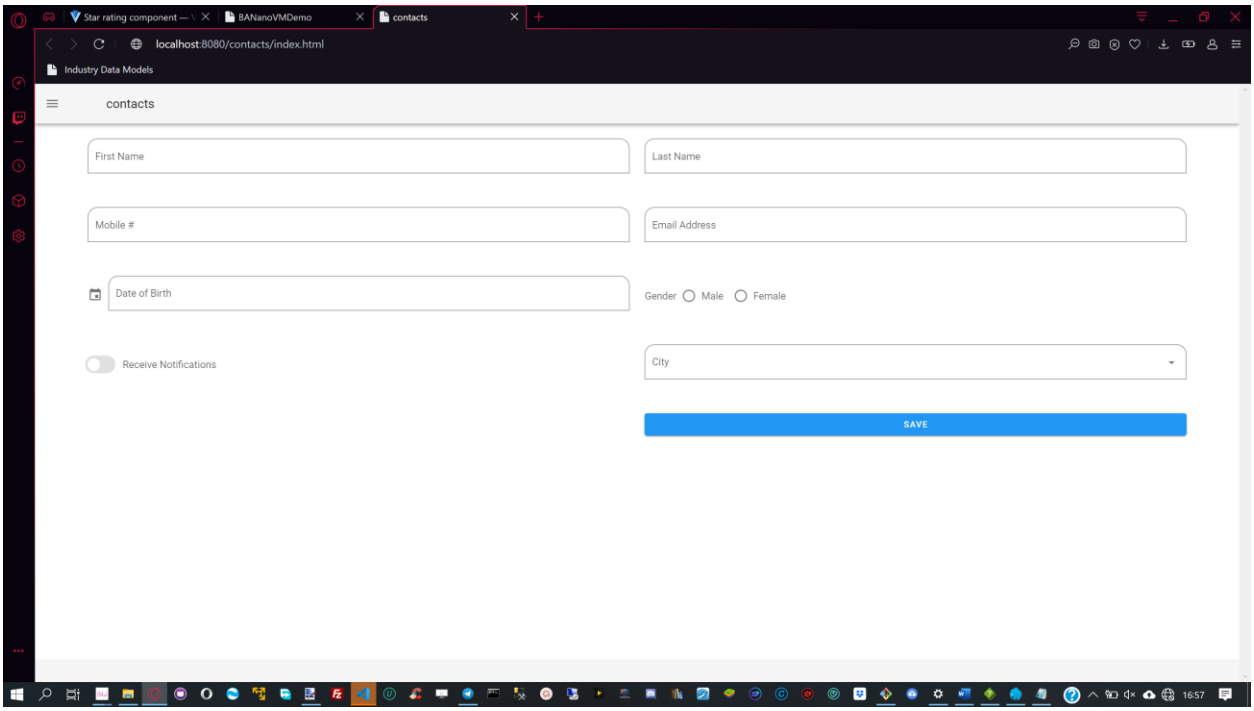
Paste this source code inside the **OurContacts** subroutine in the pgIndex code module



We need to call OurContacts sub just before the Vm.UX , so let's update the Init sub to have this change.

```
Sub Init

    'initialize the page

    vm.Initialize(Me, Main.appname)

    BuildNavBar

    BuildNavDrawer

    AddPages

    '

    OurContacts

    vm.ux

End Sub
```

Compile (in release mode) the contacts web application. Your first app should run and display the screen below on the default browser.



3. Publish your app

Whilst this webapp is far from complete, for example, we have not added any events and are not having any back-end database, the publish process involves uploading the generated BANano app to your web server. We compiled our app to laragon, so we can just locate that folder and then ftp that content to our site of choice.