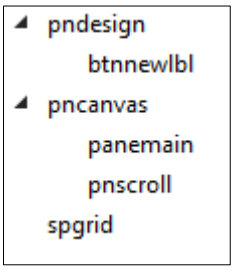
Step 3 : add a grid and labels to create a design

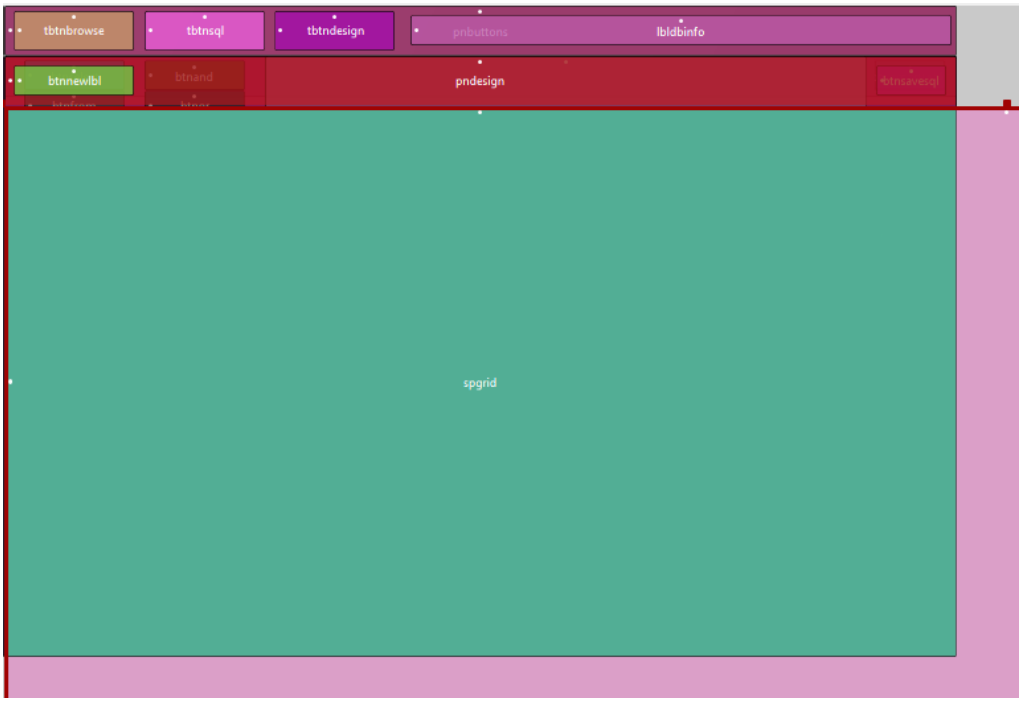
The grid is scrollable in all directions. The labels can be moved around in the grid.

Changes to the MainPage layout:

- Add a togglebutton into the pnbuttons pane, name it **tbtndesign** and set the text to “Design”. Generate the tbtndesign togglebutton member and select the SelectedChange event.



- Add a pane under the pnbuttons, name it **pndesign** and set the width at 950 and the height at 50. Set the color to white.
- Add a button into the pndesign pane, name it **btnnewlbl** and set the width at 100 and the height at 30. Set the color to a light yellow.
- Add a pane under the pndesign pane, name it **pncanvas** and set the width at 2000 and the height at 2000. The upper left corner of the pncanvas pane is aligned with the pndesign pane and right under it. Set the color to white.
- Add a pane into the pncanvas pane, name it **panemain** and set the width at 2000 and the height at 2000. Set the color to transparent.
- Add a pane into the pncanvas pane, name it **pnscrol** and set the width at 2000 and the height at 2000. Set the color to transparent.
- Add a scrollpane on top of the pncanvas pane, name it **spgrid** and set the width at 950 and the height at 600. Make sure the pannable property is checked! The spgrid should be the last view in the list!
- Uncheck the visible property of the pndesign, pncanvas and spgrid.
- Generate pndesign, btnnewlbl, pncanvas, panemain, pnscroll and spgrid. Select the Click event from the btnnewlbl button.



Changes in the B4XMainPage page:

- Make the variables Public (this will remove the warnings)

```
Public tbtndesign As ToggleButton
Public pndesign As Pane
Public btnnewlbl As Button
Public pncanvas As Pane
Public panemain As Pane
Public pnscroll As Pane
Public spgrid As ScrollPane
```

- Create a region design to keep the design subroutines together. Move the tbtndesign_SelectedChange subroutine in that region.
- Add code to the tbtndesign_SelectedChange subroutine:
    - An if test for the Selected state of the togglebutton.
    - Set pndesign, pncanvas and spgrid to visible = true
    - Set the innernode of the scrollpane spgrid to the pncanvas pane
    - Set the innernode preferred height of the scrollpane spgrid to the pncanvas preferred height
    - Send the pnscroll pane to the back
    - Call the subroutine draw_grid
    - If the selected state is false then set the pndesign, pncanvas and spgrid to visible = false

```
Private Sub tbtndesign_SelectedChange(Selected As Boolean)
    If Selected Then
        pndesign.Visible = True
        pncanvas.Visible = True
        spgrid.Visible = True
        spgrid.InnerNode = pncanvas
        spgrid.InnerNode.PrefHeight = pncanvas.prefHeight
        pnscroll.As(B4XView).SendToBack
        draw_grid
    Else
        pndesign.Visible = False
        pncanvas.Visible = False
        spgrid.Visible = False
    End If
End Sub
```

- Add the draw_grid subroutine (pages of 27 rows and 40 columns with a gridsize of 20 pixels)
    - Add a gridsize integer and set it to 20

```
Public gridsize As Int = 20
```

    - Set the pagewidth variable to 800 and the pageheight variable to 540. This will show the black lines indicating the page limits.
    - Don’t forget to invalidate the scrollcanvas!

```
Private Sub draw_grid
    Dim pagewidth As Int = 800
    Dim pageheight As Int = 540
    Dim scrollcanvas As B4XCanvas
    scrollcanvas.Initialize(pnscroll)
    ' horizontal lines
    For ypos = 0 To pnscroll.prefHeight Step gridsize
        If ypos Mod pageheight = 0 Then

    scrollcanvas.DrawLine(0,ypos,pnscroll.prefWidth,ypos,xui.Color_Black,2)
        Else

    scrollcanvas.DrawLine(0,ypos,pnscroll.prefWidth,ypos,xui.Color_Gray,0.5)
        End If
    Next
    ' vertical lines
    For xpos = 0 To pnscroll.prefWidth Step gridsize
        If xpos Mod pagewidth = 0 Then

    scrollcanvas.DrawLine(xpos,0,xpos,pnscroll.PrefHeight,xui.Color_Black,2)
        Else

    scrollcanvas.DrawLine(xpos,0,xpos,pnscroll.PrefHeight,xui.Color_Gray,0.5)
        End If
    Next
    scrollcanvas.Invalidate
End Sub
```

- Add the snap_to_grid subroutine
  - Add a currnode variable to the Class_Gobals subroutine. This will refer to the label that was clicked on (set in the **lbl**_MouseClicked event subroutine – see below).

```
Public currnode As B4XView
```

  - If the view to snap is not the current node then return from the subroutine
  - Set the x offset and test if it's less than or equal to 10 pixels and adjust the current node left value
  - Set the y offset and test if it's less than or equal to 10 pixels and adjust the current node top value
  - Add tests to snap the current node within the range of 0 to 2000.

```
Public Sub snap_to_grid(vw As B4XView)
    If vw <> currnode Then Return
    If currnode.IsInitialized Then
        Dim xoff As Int = currnode.Left Mod gridsize
        If xoff <= 10 Then
            currnode.Left = currnode.Left - xoff
        Else
            currnode.Left = currnode.Left - xoff
        End If
        Dim yoff As Int = currnode.Top Mod gridsize
        If yoff <= 10 Then
```

```
                currnode.top = currnode.top - yoff
            Else
                currnode.top = currnode.top - yoff
            End If
            If currnode.Left < 0 Then currnode.Left = 0
            If currnode.Top < 0 Then currnode.Top = 0
            If currnode.Left + currnode.Width > 2000 Then currnode.Left = 2000 -
currnode.Width
            If currnode.top + currnode.Height > 2000 Then currnode.top = 2000 -
currnode.Height
        End If
End Sub
```

- Add the drag_and_drop3 variable, initialize it in the B4XPage_Created subroutine and make the drag target. Set the pane panemain as a target “txtTarget”.

```
    Public DragAndDrop3 As DragAndDrop

    …
    DragAndDrop3.Initialize(Me)
    DragAndDrop3.MakeDragTarget(panemain,"txtTarget")
```

- Add the txtTarget_DragOver and the txtTarget_DragDropped subroutines
  - Set the transfer mode in the txtTarget_DragOver subroutine
  - Get the string from the dragboard (Db.GetString)
  - Set and initialize a **lbl** label variable. Make sure to set eventname to “lbl”!
  - Set the text, left, top and textsize from the **lbl** view.
  - Use the cssutils to set the border, backgroundcolor and padding style of the **lbl** view
  - Add the **lbl** node to the panemain pane using the left, top, width and height properties of the **lbl** view
  - Set the drop completed to true to finalize the dragdropped event

```
Sub txtTarget_DragOver(e As DragEvent)

    If e.GetDragboard.HasString Then e.AcceptTransferModes(TransferMode.MOVE)
End Sub
Sub txtTarget_DragDropped(e As DragEvent)
    Dim Db As Dragboard = e.GetDragboard
    Dim lbl As Label
    lbl.Initialize("lbl")
    If Db.HasString Then
        Log(Db.GetString)
        lbl.Text = Db.GetString
        lbl.Left = e.X
        lbl.Top = e.Y
        Dim xoff As Int = e.X Mod gridsize
        If xoff <= 10 Then
            lbl.Left = lbl.Left - xoff
        Else
            lbl.Left = lbl.Left - xoff
```

```
            End If
            Dim yoff As Int = e.Y Mod gridsize
            If yoff <= 10 Then
                  lbl.top = lbl.top - yoff
            Else
                  lbl.top = lbl.top - yoff
            End If
            lbl.TextSize = 15
            CSSUtils.SetBorder(lbl,1,fx.Colors.Blue,5)
            CSSUtils.SetBackgroundColor(lbl,fx.Colors.ARGB(255,216,238,255))
            CSSUtils.SetStyleProperty(lbl,"-fx-padding","3 3 3 3")
            panemain.AddNode(lbl,lbl.Left,lbl.Top,lbl.Width,lbl.Height)
            e.SetDropCompleted(True)
            Return
      End If
End Sub
```

- Add the label "lbl" mouse events subroutines
  - Set pressedX, pressedY and mousedrag variables in the Class_Globals subroutine

```
Public pressedX As Double
Public pressedY As Double
Public mousedrag As Boolean
```

  - In the lbl_MouseClicked subroutine the current node variable is set
  - In the lbl_MouseDragged subroutine the left and top position of the node (a label in this case) is adjusted to where the mouse pointer is. The mousedrag variable is set to true.
  - The mouse button that was used is tested in the lbl_MousePressed subroutine. When the right button was used the label is removed from the parent (panemain). There is no "are you sure" question at this point!
  - In the lbl_MouseReleased subroutine the snap_to_grid subroutine is called to snap the label to the grid
  - In the subroutines lbl_MouseEntered and lbl_MouseExited the mousedrag variable is set to false
  - The lbl_MouseMoved subroutine is not used (yet).

```
Private Sub lbl_MouseClicked(EventData As MouseEvent)
      Dim view As B4XView = Sender
      currnode = view
      EventData.Consume
End Sub
Private Sub lbl_MouseDragged (EventData As MouseEvent)
      Dim nod As Node = Sender
      Dim event As JavaObject = EventData
      Dim offsetX, offsetY As Double
      offsetX = event.RunMethod("getSceneX",Null) - pressedX
      offsetY =  event.RunMethod("getSceneY",Null)- pressedY
      nod.Left = nod.Left + offsetX
      nod.Top = nod.Top + offsetY
      pressedX = event.RunMethod("getSceneX",Null)
      pressedY = event.RunMethod("getSceneY",Null)
```

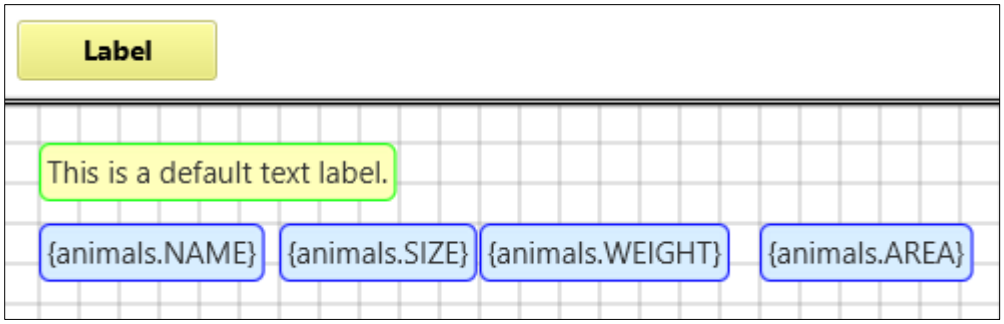
```
        currnode = nod
        EventData.Consume
        mousedrag = True
End Sub
Private Sub lbl_MousePressed (EventData As MouseEvent)
        Dim view As B4XView = Sender
        If EventData.SecondaryButtonDown Then
                view.RemoveViewFromParent
                EventData.Consume
                Return
        End If
        Dim event As JavaObject = EventData
        pressedX = event.RunMethod("getSceneX",Null)
        pressedY = event.RunMethod("getSceneY",Null)
        EventData.Consume
End Sub
Private Sub lbl_MouseReleased (EventData As MouseEvent)
        Dim view As B4XView = Sender
        snap_to_grid(view)
        EventData.Consume
End Sub
Private Sub lbl_MouseEntered (EventData As MouseEvent)
'       Dim view As B4XView = Sender
        mousedrag = False
End Sub
Private Sub lbl_MouseExited (EventData As MouseEvent)
'       Dim view As B4XView = Sender
        If mousedrag = True Then
                mousedrag = False
        End If
End Sub
Private Sub lbl_MouseMoved (EventData As MouseEvent)
'       Dim view As B4XView = Sender
End Sub
```

- Add code in the btnnewlbl_Click event subroutine.
    - Set and initialize a local **lbl** label variable. The eventname is also “lbl”!
    - Set the text, textsize, wraptext, alignment and tag properties
    - Use the cssutils to set the padding style, border and backgrount color properties
    - Add the lbl node to the panemain pane

```
Private Sub btnnewlbl_Click
        Dim lbl As Label
        lbl.Initialize("lbl")
        lbl.Text = "This is a default text label. "
        lbl.TextSize = 15
        lbl.WrapText = True
        lbl.Alignment = "TOP_LEFT"
        lbl.Tag = lbl.Text
        CSSUtils.SetStyleProperty(lbl,"-fx-padding","3 3 3 3")
```

```
    CSSUtils.SetBorder(lbl,1,fx.Colors.Green,5)
    CSSUtils.SetBackgroundColor(lbl,fx.Colors.Yellow)
    panemain.AddNode(lbl,0,0,lbl.prefWidth,lbl.prefHeight)
End Sub
```

You can run the application now and test the labels and the movements of the grid and labels. Drag and drop a database field label on the grid and see how it looks like.



- Make a new layout for a label dialog
  - The Label1 contains the text “Label text:”
  - Set **talabeltext** textarea width to 500 and the height to 200. Uncheck the wrap text and leave the Editable check on.
  - Generate the member **talabeltext**.
  - Save the layout with the name “labeldialog_layout”

- Add code to use a label dialog and a textarea to provide a specific text to show in the label
  - Make the variable **talabeltext** Public and add a **labeldialog** variable
    ```
    Public talabeltext As TextArea
    Public labeldialog As B4Xdialog
    ```
  - Set and initialize the **labeldialog** variable in the B4XPage_Created subroutine.
    ```
    labeldialog.Initialize(Root)
    labeldialog.PutAtTop = True
    labeldialog.BackgroundColor = xui.Color_LightGray
    labeldialog.BodyTextColor = xui.Color_Blue
    ```
  - **IMPORTANT NOTE**: add an empty text file to the Files Manager and name it “Dialog.css”. This will fix the issue with a B4Xdialog and textarea’s.

- Add code to the btnnewlbl_Click subroutine as the first lines in the subroutine. Set a title variable, a title_color variable and a pane variable. Call the setup_dialog subroutine to fill the pane. Use the ShowCustom method of the labeldialog and wait for the result.

```
Dim title As String = " Add new label"
Dim title_color As Paint = fx.Colors.ARGB(255,186,255,158)
Dim pnl1 As Pane = setup_dialog("labeldialog_layout",title,title_color,300)
    talabeltext.Text = ""
Dim rsub1 As ResumableSub =  labeldialog.ShowCustom(pnl1, "OK", "", "CANCEL")
Wait For (rsub1) Complete (Result As Int)
If Result = xui.dialogResponse_Positive Then
```

- Under the test for a positive result you can keep the remaining code from the btnnewlbl_Click subroutine. Add a End If below that code.
- Change the line lbl.Text = "This is a default text label." to lbl.Text = talabeltext.Text
- Add the setup_dialog subroutine that returns a pane with a title label.

```
Sub setup_dialog(strlayout As String,lbltext As String,lblcolor As Paint,height As Int) As Pane
    Dim pnl As B4XView = xui.CreatePanel("")
    pnl.SetLayoutAnimated(0,0,0,520,height)
    pnl.LoadLayout(strlayout)
    Dim lbl As Label
    lbl.Initialize("")
    lbl.Text = lbltext
    lbl.TextSize = 18
    lbl.TextColor = fx.Colors.From32Bit(0xFF000000)
    CSSUtils.SetBackgroundColor(lbl,lblcolor)
    pnl.AddView(lbl,0dip,0dip,520,30dip)
    Return pnl
End Sub
```

You can run the application again to test everything.

In the next step (or steps) more code is added to use report tags in the design grid and to scan the grid and create word documents…