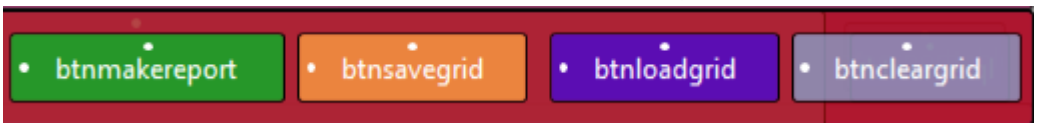
Step 6 : save and load the grid labels, more code for image processing.

Changes in the layout:

- Add the buttons **btnsavegrid**, **btnloadgrid** and **btncleargrid** into the pndesign pane. For each button the width is 100 and the height is 30.



- Generate the members and check the Click event from each button.

Changes in the B4XMainPage page:

- Add code to the btnsavegrid_Click subroutine to save the gridmap.
  The text file gets a *.map extension.
  First the gridmap is filled with the current content of the panemain pane (in the fill_gridmap subroutine). Then the gridmap map is saved in the file.
  Note: each key value pair is a single line string. The tagdata values of the labels are converted to strings.

```
Private Sub btnsavegrid_Click
	fill_gridmap
	fchoose.InitialDirectory = xui.DefaultFolder
	fchoose.setExtensionFilter("Map", Array As String("*.map"))
	Dim fname As String =
fchoose.ShowSave(B4XPages.GetNativeParent(B4XPages.MainPage))
	If fname = "" Then Return
	Dim path As String = fname.SubString2(0,fname.LastIndexOf("\")+1)
	Dim filename As String = fname.SubString(fname.LastIndexOf("\")+1)
	File.WriteMap(path,filename,gridmap)
	Log("saved gridmap: " & gridmap)
	xui.MsgboxAsync("File " & fname & " saved.","Save Map")
End Sub
```

- Add code to the btnloadgrid_Click subroutine to load the gridmap from the saved file. The extension is set to *.map. The gridmap is filled with the strings from the file. In a for each loop from the gridmap keys the tagdata string is converted to the tagdata type using the set_tagdata_from_string subroutine.
  A new label is created and added to the panemain pane. The label gets the correct color settings.

```
Private Sub btnloadgrid_Click
	fchoose.InitialDirectory = xui.DefaultFolder
	fchoose.setExtensionFilter("Map", Array As String("*.map"))
	Dim fname As String =
fchoose.ShowOpen(B4XPages.GetNativeParent(B4XPages.MainPage))
	If fname = "" Then Return
	Dim path As String = fname.SubString2(0,fname.LastIndexOf("\")+1)
	Dim filename As String = fname.SubString(fname.LastIndexOf("\")+1)
	gridmap.Clear
	Dim gridmap As Map = File.ReadMap(path,filename)
'	Log(gridmap)
```

```
        panemain.RemoveAllNodes
        For Each key In gridmap.Keys
                Dim rowcol As String = key
                Dim tdstring As String = gridmap.Get(key)
                Dim td As tagdata = set_tagdata_from_string(tdstring)
                Dim lbl As Label
                lbl.Initialize("lbl")
                lbl.Text = td.text
                lbl.TextSize = 15
                lbl.WrapText = True
                lbl.Alignment = "TOP_LEFT"
                lbl.TooltipText = lbl.text
                lbl.Tag = lbl.Text
                CSSUtils.SetStyleProperty(lbl,"-fx-padding","3 3 3 3")
                Select td.labeltype
                        Case "FLD"
                                CSSUtils.SetBorder(lbl,1,fx.Colors.Blue,5)
                                CSSUtils.SetBackgroundColor(lbl,
fx.Colors.ARGB(255,216,238,255))
                        Case "TXT"
                                CSSUtils.SetBorder(lbl,1,fx.Colors.Green,5)
                                CSSUtils.SetBackgroundColor(lbl,
fx.Colors.ARGB(255,255,255,187))
                        Case "FCO"
                                CSSUtils.SetBorder(lbl,1,fx.Colors.Black,5)
                                CSSUtils.SetBackgroundColor(lbl,
fx.Colors.ARGB(255,255,200,200))
                        Case "IMG"
                                CSSUtils.SetBorder(lbl,1,fx.Colors.Blue,5)
                                CSSUtils.SetBackgroundImage(lbl,td.imgdir,td.imgfile)
                                lbl.PrefWidth = td.numcols * gridsize
                                lbl.PrefHeight = td.numrows * gridsize
                        Case Else
                                CSSUtils.SetBorder(lbl,1,fx.Colors.Black,5)
                                CSSUtils.SetBackgroundColor(lbl,
fx.Colors.ARGB(255,202,255,187))
                End Select
                lbl.Top = rowcol.SubString2(0,rowcol.IndexOf("|")) * gridsize
                lbl.Left = rowcol.SubString(rowcol.IndexOf("|")+1) * gridsize
                lbl.Text = td.text
                lbl.Tag = td
                panemain.AddNode(lbl,lbl.left,lbl.top,lbl.prefWidth,lbl.prefHeight)
        Next
End Sub
```

- Add the set_tagdata_from_string subroutine. A list (tdlist) is used to get the tagdata pieces from the tagdata string (tdstring). The pieces are then assigned to the tagdata properties using the part after the = sign. Trailing spaces or control characters are trimmed off.

```
Private Sub set_tagdata_from_string(tdstring As String) As tagdata
    Dim td As tagdata
    td.Initialize
    Dim tdlist As List = Regex.Split(",",tdstring)
    Dim strlabeltype As String = tdlist.Get(1)
    td.labeltype = strlabeltype.SubString(strlabeltype.IndexOf("=")+1).Trim
    Dim strrow As String = tdlist.Get(2)
    td.row = strrow.SubString(strrow.IndexOf("=")+1).Trim
    Dim strcol As String = tdlist.Get(3)
    td.col = strcol.SubString(strcol.IndexOf("=")+1).Trim
    Dim strnumrows As String = tdlist.Get(4)
    td.numrows = strnumrows.SubString(strnumrows.IndexOf("=")+1).Trim
    Dim strnumcols As String = tdlist.Get(5)
    td.numcols = strnumcols.SubString(strnumcols.IndexOf("=")+1).Trim
    Dim strimgdir As String = tdlist.Get(6)
    td.imgdir = strimgdir.SubString(strimgdir.IndexOf("=")+1).Trim
    Dim strimgfile As String = tdlist.Get(7)
    td.imgfile = strimgfile.SubString(strimgfile.IndexOf("=")+1).Trim
    Dim strdbfile As String = tdlist.Get(8)
    td.dbfile = strdbfile.SubString(strdbfile.IndexOf("=")+1).Trim
    Dim strdbpath As String = tdlist.Get(9)
    td.dbpath = strdbpath.SubString(strdbpath.IndexOf("=")+1).Trim
    Dim strdbname As String = tdlist.Get(10)
    td.dbname = strdbname.SubString(strdbname.IndexOf("=")+1).Trim
    Dim strtblname As String = tdlist.Get(11)
    td.tblname = strtblname.SubString(strtblname.IndexOf("=")+1).Trim
    Dim strfldname As String = tdlist.Get(12)
    td.fldname = strfldname.SubString(strfldname.IndexOf("=")+1).Trim
    Dim strsql As String = tdlist.Get(13)
    td.strsql = strsql.SubString(strsql.IndexOf("=")+1).Trim
    Dim strtext As String = tdlist.Get(14)
    strtext = strtext.SubString2(0,strtext.Length-1).Trim
    td.text = strtext.SubString(strtext.IndexOf("=")+1).Trim
    If td.text = "" Then td.text = TAB                ' simulate an empty paragraph
    Return td
End Sub
```

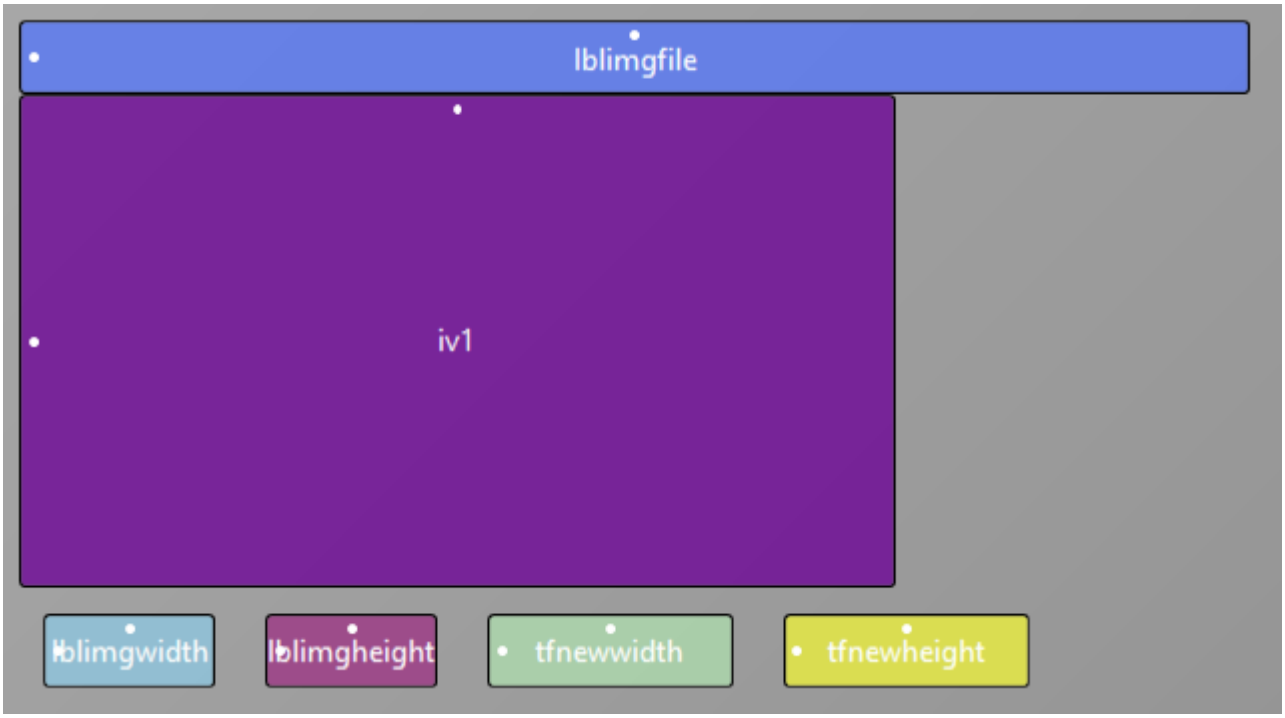
- Add code to the btncleargrid_Click subroutine. All nodes will be removed from the panemain pane.

```
Private Sub btncleargrid_Click
    panemain.RemoveAllNodes
End Sub
```

Changes for the image processing:

- Add a imagedialog in the layouts and save it as “imagedialog_layout”. The label **lblimagefile** has a width of 500, a height of 30 and a top of 50. The ImageView **iv1** has a width of 356 and a height of 200. The labels **lblimgwidth** and **lblimgheight** have a width of 70 and a height of 30. The TextFields **tfnewwidth** and **tfnewheight** have a width of

100 and a height of 30. Set the prompts to “New width” and “New height”.



- Generate the members and set the variables to public.
- Add code for the imagedialog: declare a imagedialog variable in the Class_Globals subroutine, initialize the imagedialog in the B4XPage_Created subroutine.

```
Public imagedialog As B4Xdialog
imagedialog.Initialize(Root)
imagedialog.PutAtTop = True
imagedialog.BackgroundColor = xui.Color_LightGray
imagedialog.BodyTextColor = xui.Color_Blue
```

- Change the btnnewimage_Click subroutine to the following code:
  After a file is selected the imagedialog appears. The image width and height are displayed and you can change the width and height in the textfields. The default width and height is set at 100.

```
Private Sub btnnewimage_Click
	fchoose.SetExtensionFilter("Image files",Array As String("*.jpg","*.png"))
	Dim filename As String =
fchoose.ShowOpen(B4XPages.GetNativeParent(B4XPages.MainPage))
	If filename <> "" Then
		Dim fpath As String = filename.SubString2(0,filename.LastIndexOf("\"))
		Dim fname As String = filename.SubString(filename.LastIndexOf("\")+1)
		Dim title As String = " Add new image"
		Dim title_color As Paint = fx.Colors.ARGB(255,186,255,158)
		Dim pnl1 As Pane = setup_dialog("imagedialog_layout",title,title_color,340)
		lblimgfile.Text = fpath & "\" & fname
		Dim img As Image
		img.Initialize(fpath,fname)
	'	iv1.Initialize("")
		iv1.SetImage(img)
		lblimgwidth.Text = img.Width
		lblimgheight.Text = img.Height
		tfnewwidth.Text = ""
```

```
            tfnewheight.Text = ""
            Dim rsub1 As ResumableSub =  labeldialog.ShowCustom(pnl1, "OK", "",
"CANCEL")
            Wait For (rsub1) Complete (Result As Int)
            If Result = xui.dialogResponse_Positive Then
                If tfnewwidth.Text = "" Then tfnewwidth.Text = "100"
                If tfnewheight.Text = "" Then tfnewheight.Text = "100"
                Dim lbl As Label
                lbl.Initialize("lbl")
                lbl.Text = ""
                lbl.Tag = lbl.Text
                lbl.prefWidth = tfnewwidth.Text
                lbl.prefHeight = tfnewheight.Text
                CSSUtils.SetBorder(lbl,1,fx.Colors.Blue,5)
                CSSUtils.SetBackgroundImage(lbl,fpath,fname)
                Dim td As tagdata
                td.Initialize
                td.row = lbl.Top/gridsize
                td.col = lbl.Left/gridsize
                td.numrows = lbl.PrefHeight / 20
                td.numcols = lbl.PrefWidth / 20
                td.imgdir = fpath
                td.imgfile = fname
                td.labeltype = "IMG"
                td.text = ""
                lbl.Tag = td
                panemain.AddNode(lbl,0,0,lbl.prefWidth,lbl.prefHeight)
            End If
        End If
End Sub
```

- Add code to the lbl_MouseClicked after the test tdl.labeltype = “TXT”. If you click on an image you can change the settings from that image with the change_image_settings subroutine.

```
If tdl.labeltype = "IMG" Then
        change_image_settings
End If
```

- Add the change_image_settings subroutine.

```
Private Sub change_image_settings
        Dim title As String = " Change image settings"
        Dim title_color As Paint = fx.Colors.ARGB(255,186,255,158)
        Dim pnl1 As Pane = setup_dialog("imagedialog_layout",title,title_color,340)
        Dim tdi As tagdata = currnode.Tag
        lblimgfile.Text = tdi.imgdir & "\" & tdi.imgfile
        Dim img As Image
        img.Initialize(tdi.imgdir,tdi.imgfile)
        '       iv1.Initialize("")
        iv1.SetImage(img)
        lblimgwidth.Text = img.Width
```

```
        lblimgheight.Text = img.Height
        tfnewwidth.Text = tdi.numcols * gridsize
        tfnewheight.Text = tdi.numrows * gridsize
        Dim rsub1 As ResumableSub =  labeldialog.ShowCustom(pnl1, "OK", "",
"CANCEL")
        Wait For (rsub1) Complete (Result As Int)
        If Result = xui.dialogResponse_Positive Then
                Dim lbl As Label
                lbl.Initialize("lbl")
                lbl.Text = ""
                lbl.Tag = currnode.Tag
                lbl.Top = currnode.Top
                lbl.Left = currnode.Left
                lbl.prefWidth = tfnewwidth.Text
                lbl.prefHeight = tfnewheight.Text
                CSSUtils.SetBorder(lbl,1,fx.Colors.Blue,5)
                CSSUtils.SetBackgroundImage(lbl,tdi.imgdir,tdi.imgfile)
                Dim td As tagdata
                td.Initialize
                td.row = lbl.Top/gridsize
                td.col = lbl.Left/gridsize
                td.numrows = lbl.PrefHeight / 20
                td.numcols = lbl.PrefWidth / 20
                td.imgdir = tdi.imgdir
                td.imgfile = tdi.imgfile
                td.labeltype = "IMG"
                td.text = ""
                lbl.Tag = td
                currnode.RemoveViewFromParent
                panemain.AddNode(lbl,0,0,lbl.prefWidth,lbl.prefHeight)
        End If
End Sub
```

- In the set_one_row_table subroutine you need to change the case “IMG”

```
Case "IMG"
        Dim imgwidth As Int = (td.numcols+1)*gridsize
        Dim imgheight As Int = (td.numrows+1)*gridsize
        sb.Append($"[cell width=${imgwidth+20}][p alignment=center][img
dir="${td.imgdir}" filename="${td.imgfile}" width=${imgwidth} height=${imgheight}
/][/p][/cell]"$)
```

- In the write_paragraph subroutine you also need to change the case “IMG”

```
Case "IMG"
        Dim imgwidth As Int = (td.numcols+1)*gridsize
        Dim imgheight As Int = (td.numrows+1)*gridsize
        sb.Append($"[p Indentationleft=${indent}][img dir="${td.imgdir}"
filename="${td.imgfile}" width=${imgwidth} height=${imgheight} /][/p]"$)
```

- The btnloadgrid_Click subroutine already contains the code for the width and height adjustments of an image.

The imagedialog looks like this: