ConstraintLayout

Library

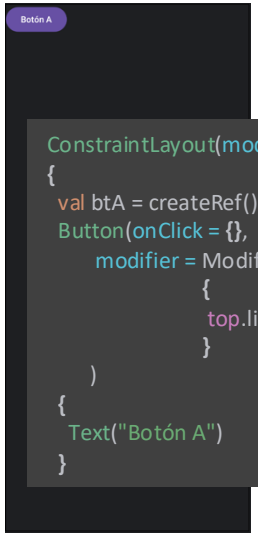
group-id: "androidx.constraintlayout"

artifact-id: "constraintlayout-compose"

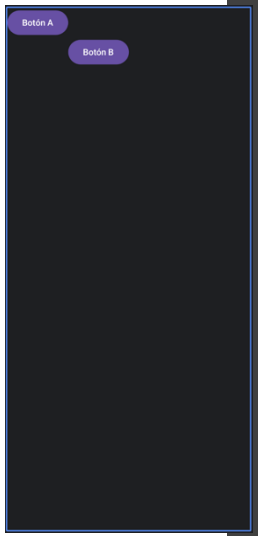
version: "1.1.0"

ConstraintLayout

Align top edges



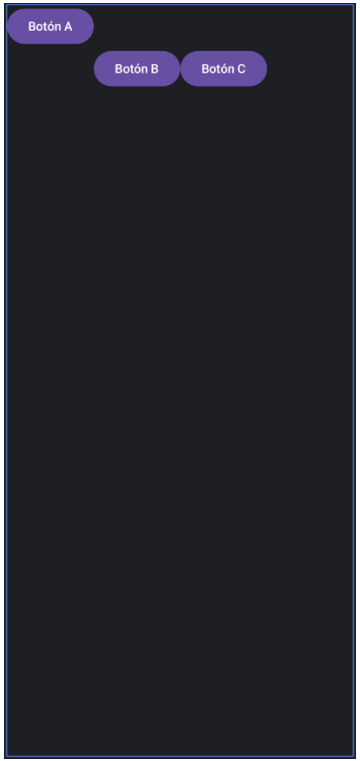

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
 val btA = createRef()
 Button(onClick = {},
      modifier = Modifier.constrainAs(btA)
                 {
                  top.linkTo(parent.top)
                 }
    )
 {
  Text("Botón A")
 }
```




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
 val (btA, btB) = createRefs()
 Button(onClick = {},
      modifier = Modifier.constrainAs(btA)
                 {
                  top.linkTo(parent.top)
                 }
    )
    {
      Text("Botón A")
    }

 Button(onClick = {},
      modifier = Modifier.constrainAs(btB)
                 {
                  top.linkTo(btA.bottom)
                 }
    )
    {
      Text("Botón B")
    }
}
```

ConstraintLayout

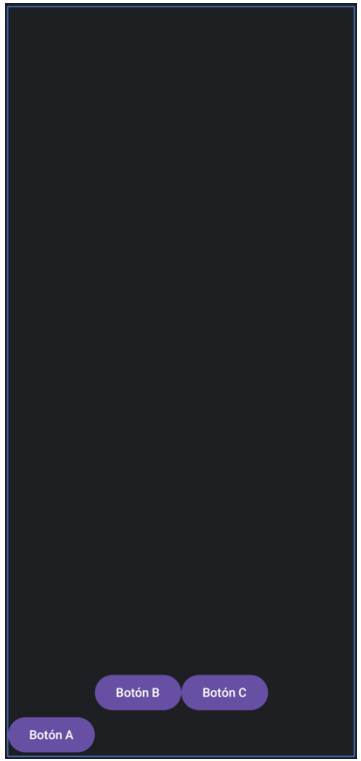
Align top edges




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()

   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                {
                 top.linkTo(parent.top)
                }
        ) { Text("Botón A") }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                {
                 top.linkTo(btA.bottom)

                }
        ) { Text("Botón B") }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                {
                 top.linkTo(btB.top)

                }
        ) { Text("Botón C") }
}
```

ConstraintLayout

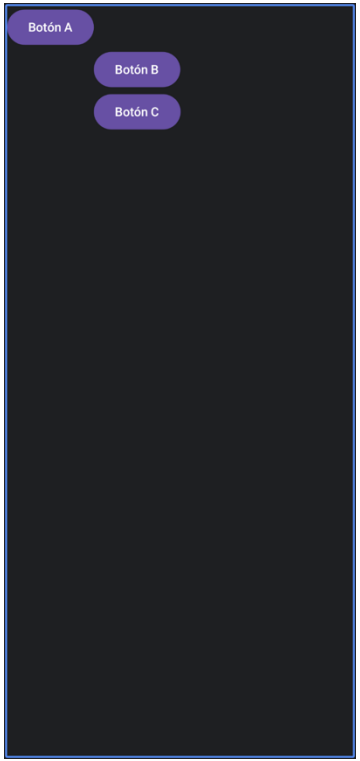
Align bottom edges




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()

   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                {
                 bottom.linkTo(parent.bottom)
                }
        ) { Text("Botón A") }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                {
                 bottom.linkTo(btA.top)

                }
        ) { Text("Botón B") }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                {
                 bottom.linkTo(btB.bottom)

                }
        ) { Text("Botón C") }
}
```

ConstraintLayout

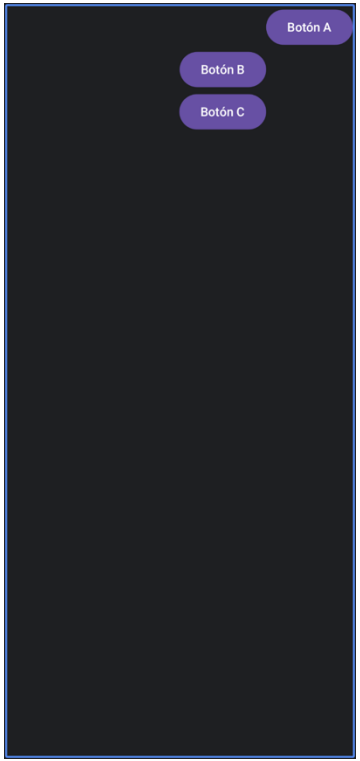
Align start edges




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                         top.linkTo(parent.top)
                         start.linkTo(parent.start)
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                         top.linkTo(btA.bottom)
                         start.linkTo(btA.end)
                        }
       ) { Text("Botón B") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btC)
                        {
                         top.linkTo(btB.bottom)
                         start.linkTo(btB.start)
                        }
       ) { Text("Botón C") }
}
```

ConstraintLayout

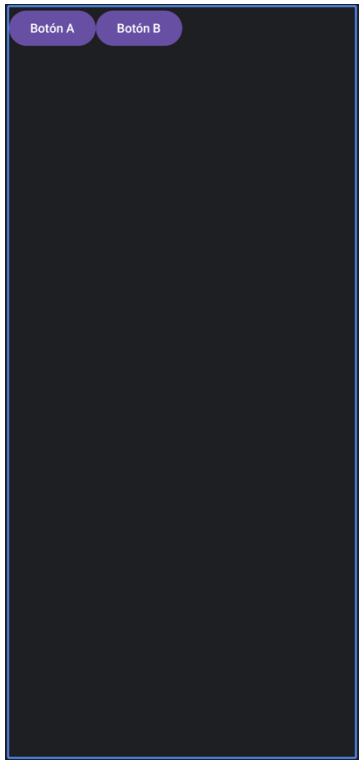
Align end edges




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC, btD) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                         top.linkTo(parent.top)
                         end.linkTo(parent.end)
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                         top.linkTo(btA.bottom)
                         end.linkTo(btA.start)
                        }
       ) { Text("Botón B") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btC)
                        {
                         top.linkTo(btB.bottom)
                         end.linkTo(btB.end)
                        }
       ) { Text("Botón C") }
}
```

ConstraintLayout

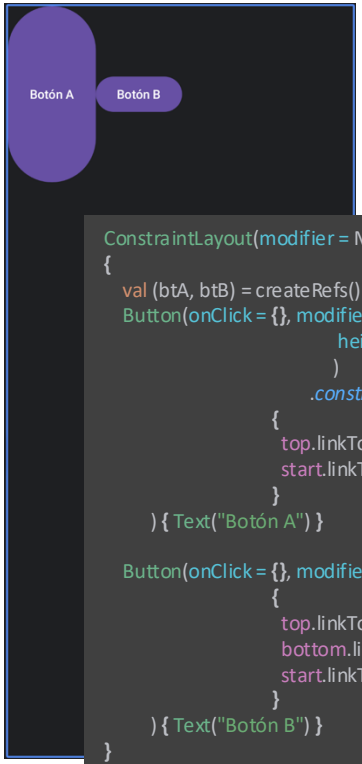
Align Baseline



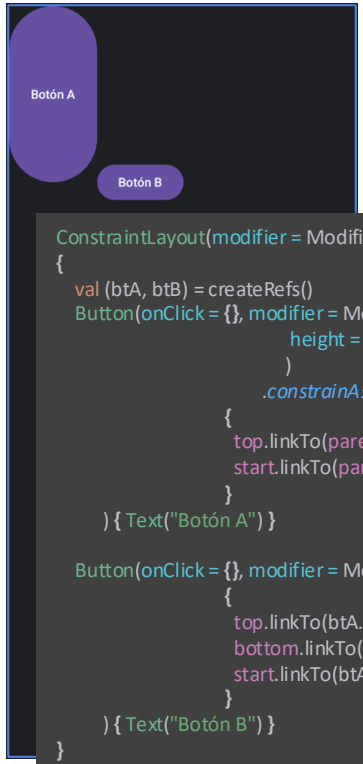

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                         top.linkTo(parent.top)
                         start.linkTo(parent.start)
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                         baseline.linkTo(btA.baseline)
                         start.linkTo(btA.end)
                        }
       ) { Text("Botón B") }
}
```

ConstraintLayout

Align vertical centers



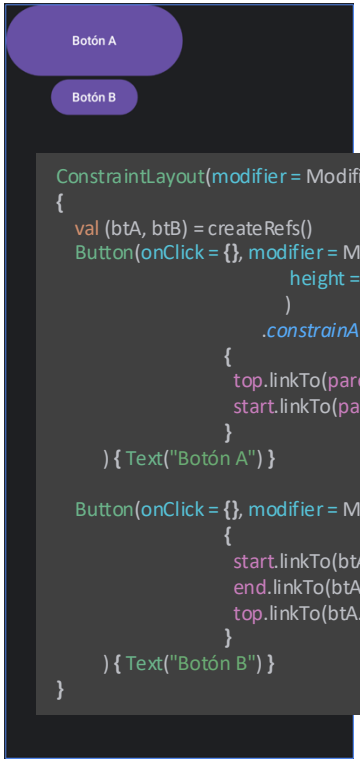

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.size(width = 100.dp,
                                    height = 200.dp
                                   )
                               .constrainAs(btA)
                          {
                           top.linkTo(parent.top)
                           start.linkTo(parent.start)
                          }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                          {
                           top.linkTo(btA.top)
                           bottom.linkTo(btA.bottom)
                           start.linkTo(btA.end)
                          }
       ) { Text("Botón B") }
}
```



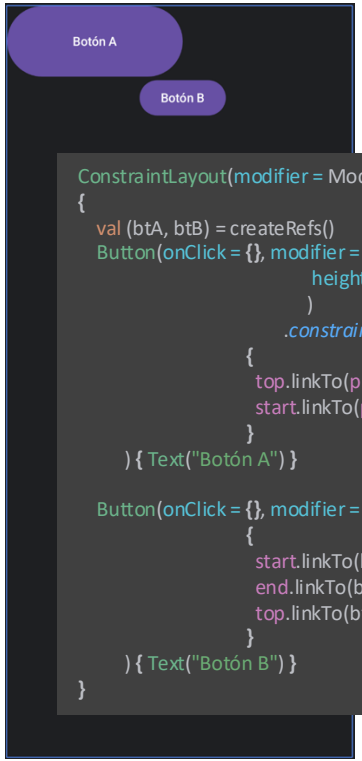

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.size(width = 100.dp,
                                    height = 200.dp
                                   )
                               .constrainAs(btA)
                          {
                           top.linkTo(parent.top)
                           start.linkTo(parent.start)
                          }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                          {
                           top.linkTo(btA.bottom)
                           bottom.linkTo(btA.bottom)
                           start.linkTo(btA.end)
                          }
       ) { Text("Botón B") }
}
```

ConstraintLayout

Align horizontal centers



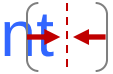

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.size(width = 200.dp,
                                  height = 80.dp
                                  )
                               .constrainAs(btA)
                          {
                           top.linkTo(parent.top)
                           start.linkTo(parent.start)
                          }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                          {
                           start.linkTo(btA.start)
                           end.linkTo(btA.end)
                           top.linkTo(btA.bottom)
                          }
       ) { Text("Botón B") }
}
```



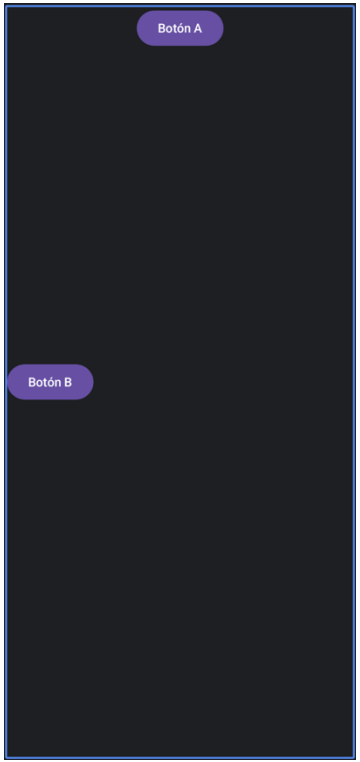

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.size(width = 200.dp,
                                  height = 80.dp
                                  )
                               .constrainAs(btA)
                          {
                           top.linkTo(parent.top)
                           start.linkTo(parent.start)
                          }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                          {
                           start.linkTo(btA.end)
                           end.linkTo(btA.end)
                           top.linkTo(btA.bottom)
                          }
       ) { Text("Botón B") }
}
```

ConstraintLayout

Center horizontally in parent



Center vertically in parent

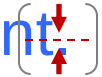


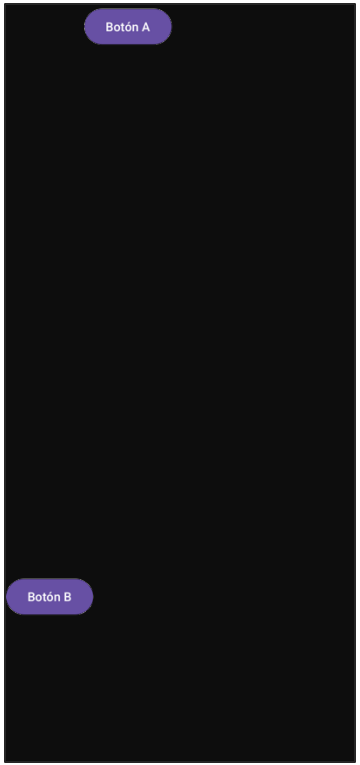

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                          top.linkTo(parent.top)
                          start.linkTo(parent.start)
                          end.linkTo(parent.end)
                          horizontalBias = 0.5f
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                          start.linkTo(parent.start)
                          top.linkTo(parent.top)
                          bottom.linkTo(parent.bottom)
                          verticalBias = 0.5f
                        }
       ) { Text("Botón B") }
}
```

ConstraintLayout

Center horizontally in parent with bias

Center vertically in parent with bias



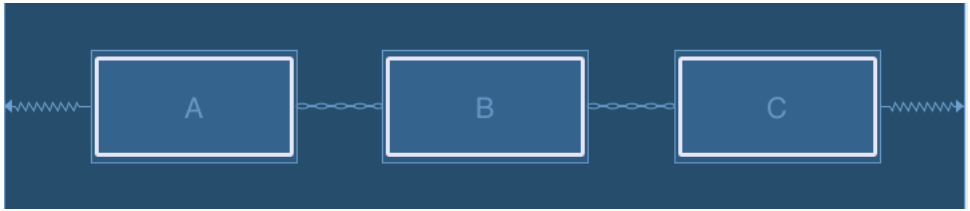



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                          top.linkTo(parent.top)
                          start.linkTo(parent.start)
                          end.linkTo(parent.end)
                          horizontalBias = 0.3f
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                          start.linkTo(parent.start)
                          top.linkTo(parent.top)
                          bottom.linkTo(parent.bottom)
                          verticalBias = 0.8f
                        }
       ) { Text("Botón B") }
}
```

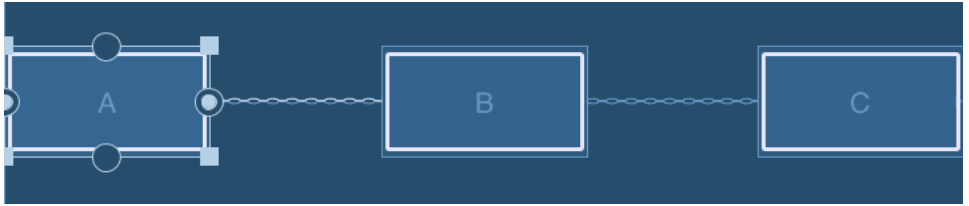
ConstraintLayout

Horizontal Chain

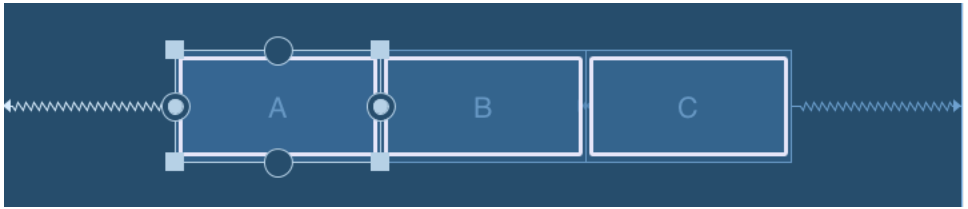
spread



spread_inside



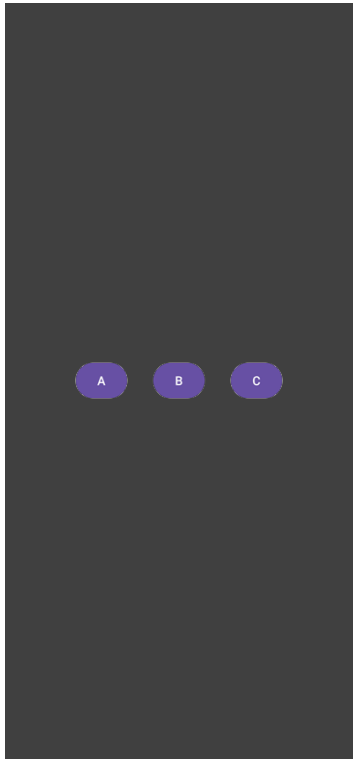
packed



ConstraintLayout

Horizontal Chain

spread

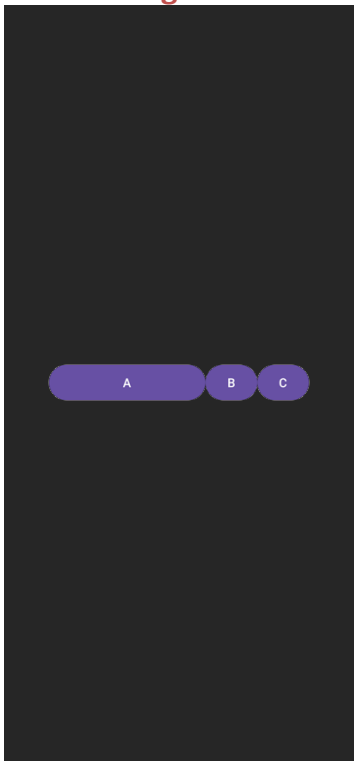



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val horizontalChain = createHorizontalChain(btA, btB, btC, chainStyle = ChainStyle.Spread)
   constrain( horizontalChain )
   {
      start.linkTo(parent.start, 50.dp)
      end.linkTo(parent.end, 50.dp)
   }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                   {
                     top.linkTo(parent.top)
                     bottom.linkTo(parent.bottom)
                   }
       ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                   {
                     baseline.linkTo(btA.baseline)
                   }
       ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                   {
                     baseline.linkTo(btA.baseline)
                   }
       ) { Text("C") }
}
```

ConstraintLayout

Horizontal Chain

spread + weight

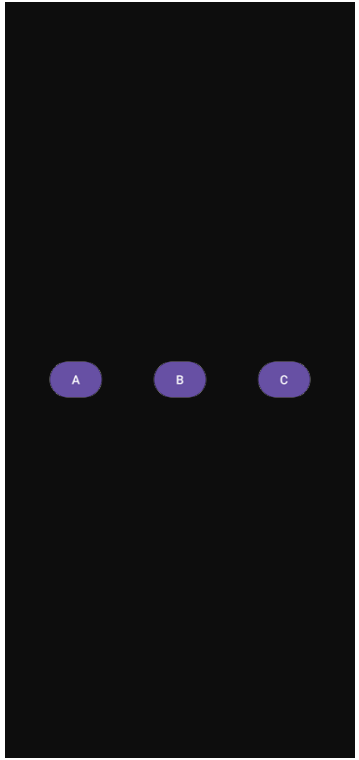



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val horizontalChain = createHorizontalChain(btA, btB, btC, chainStyle = ChainStyle.Spread)
   constrain( horizontalChain )
   {
      start.linkTo(parent.start, 50.dp)
      end.linkTo(parent.end, 50.dp)
   }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                   {
                    top.linkTo(parent.top);
                    bottom.linkTo(parent.bottom)
                    width = Dimension.fillToConstraints; horizontalChainWeight = 1f
                   }
       ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                   {
                    baseline.linkTo(btA.baseline)
                   }
       ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                   {
                    baseline.linkTo(btA.baseline)
                   }
       ) { Text("C") }
}
```

ConstraintLayout

Horizontal Chain

spread inside

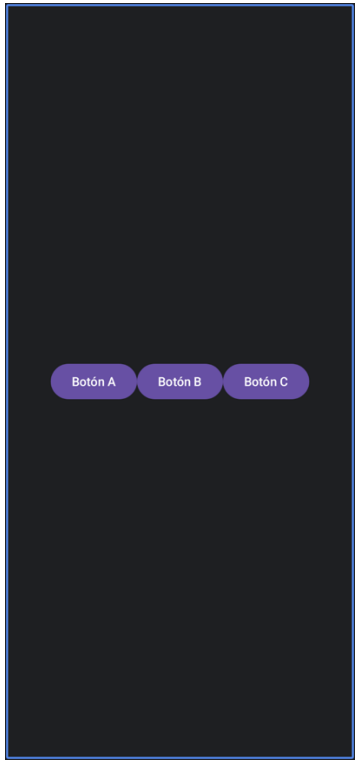



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val horizontalChain = createHorizontalChain(btA, btB, btC, chainStyle = ChainStyle.SpreadInside)
   constrain( horizontalChain )
   {
      start.linkTo(parent.start, 50.dp)
      end.linkTo(parent.end, 50.dp)
   }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                   {
                     top.linkTo(parent.top)
                     bottom.linkTo(parent.bottom)
                   }
       ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                   {
                     baseline.linkTo(btA.baseline)
                   }
       ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                   {
                     baseline.linkTo(btA.baseline)
                   }
       ) { Text("C") }
}
```

ConstraintLayout

Horizontal Chain

packed

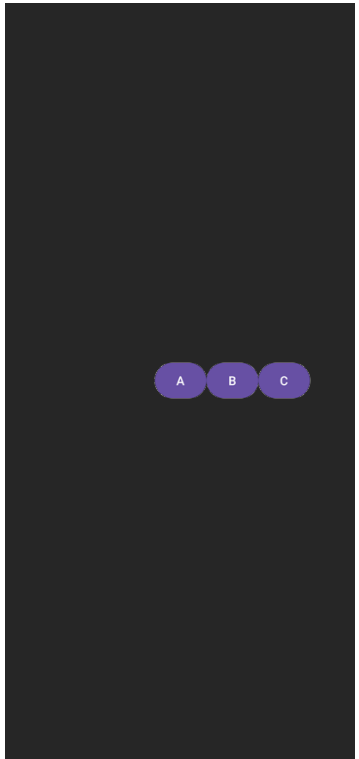



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val horizontalChain = createHorizontalChain(btA, btB, btC, chainStyle = ChainStyle.Packed)
   constrain( horizontalChain )
   {
      start.linkTo(parent.start, 50.dp)
      end.linkTo(parent.end, 50.dp)
   }

   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                   {
                     top.linkTo(parent.top)
                     bottom.linkTo(parent.bottom)
                   }
       ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                   {
                     baseline.linkTo(btA.baseline)
                   }
       ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                   {
                     baseline.linkTo(btA.baseline)
                   }
       ) { Text("C") }
}
```

ConstraintLayout

Horizontal Chain

packed + bias

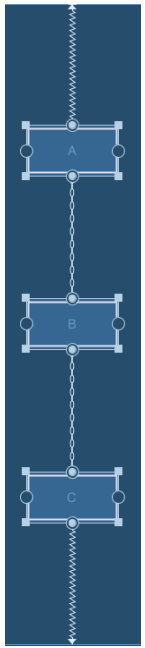



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val horizontalChain = createHorizontalChain(btA, btB, btC, chainStyle = ChainStyle.Packed)
   constrain( horizontalChain )
   {
      start.linkTo(parent.start, 50.dp)
      end.linkTo(parent.end, 50.dp)
   }
   Button(onClick = {},
         modifier = Modifier.constrainAs(btA)
                        {
                         top.linkTo(parent.top)
                         bottom.linkTo(parent.bottom)
                         horizontalBias = 1f
                        }
        ) { Text("A") }
   Button(onClick = {},
         modifier = Modifier.constrainAs(btB)
                        {
                         baseline.linkTo(btA.baseline)
                        }
        ) { Text("B") }
   Button(onClick = {},
         modifier = Modifier.constrainAs(btC)
                        {
                         baseline.linkTo(btA.baseline)
                        }
        ) { Text("C") }
}
```

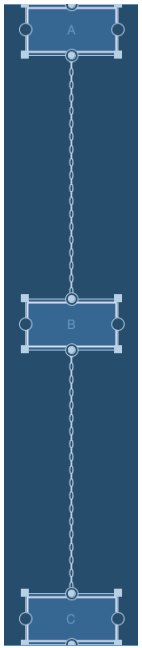
ConstraintLayout

Vertical Chain

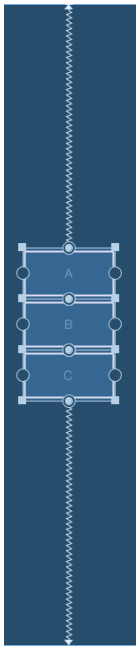
spread



spread_inside



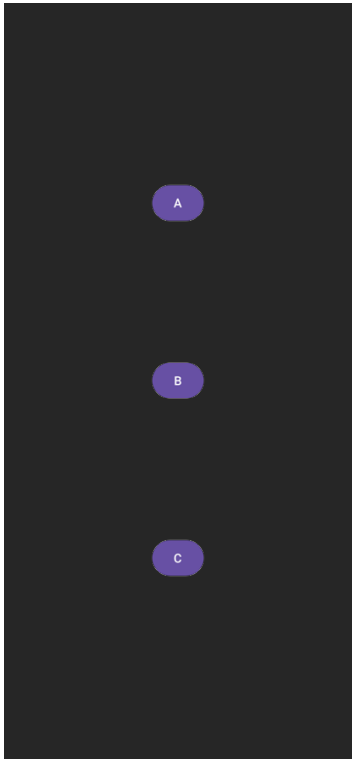
packed



ConstraintLayout

Vertical Chain

spread

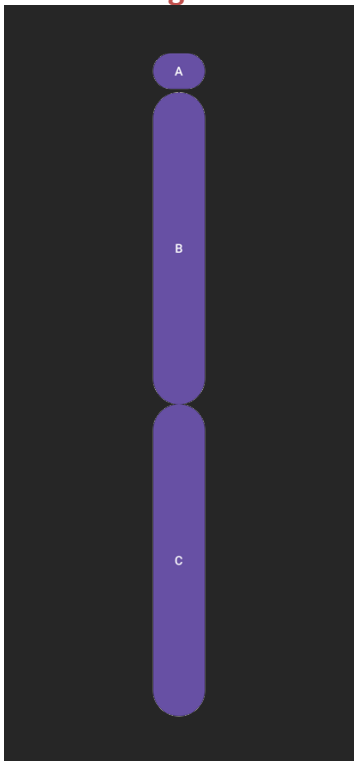



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val verticalChain = createVerticalChain(btA, btB, btC, chainStyle = ChainStyle.Spread)
   constrain( verticalChain )
   {
      top.linkTo(parent.top, 50.dp)
      bottom.linkTo(parent.bottom, 50.dp)
   }
   Button(onClick = {},
         modifier = Modifier.constrainAs(btA)
                        {
                          start.linkTo(parent.start); end.linkTo(parent.end)

                        }
        ) { Text("A") }
   Button(onClick = {},
         modifier = Modifier.constrainAs(btB)
                        {
                          start.linkTo(btA.start)
                        }
        ) { Text("B") }
   Button(onClick = {},
         modifier = Modifier.constrainAs(btC)
                        {
                          start.linkTo(btA.start)
                        }
        ) { Text("C") }
}
```

ConstraintLayout

Vertical Chain

spread + weight



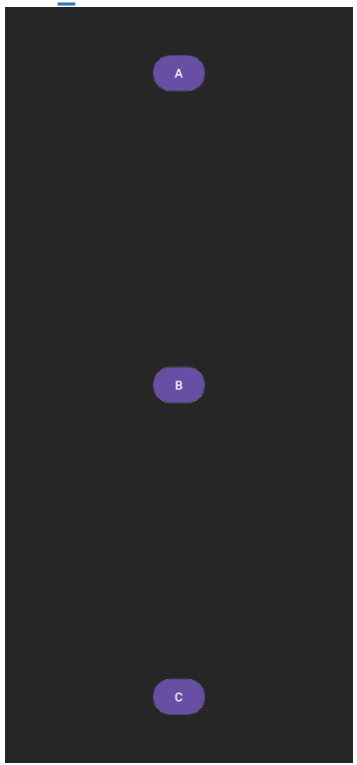

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val verticalChain = createVerticalChain(btA, btB, btC, chainStyle = ChainStyle.Spread)
   constrain( verticalChain )
   {
      top.linkTo(parent.top, 50.dp)
      bottom.linkTo(parent.bottom, 50.dp)
   }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                   {
                     start.linkTo(parent.start); end.linkTo(parent.end)
                   }
        ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                   {
                     start.linkTo(btA.start)
                     height = Dimension.fillToConstraints; verticalChainWeight = 0.5f
                   }
        ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                   {
                     start.linkTo(btA.start)
                     height = Dimension.fillToConstraints; verticalChainWeight = 0.5f
                   }
        ) { Text("C") }
}
```

ConstraintLayout

Vertical Chain

2

spread_inside

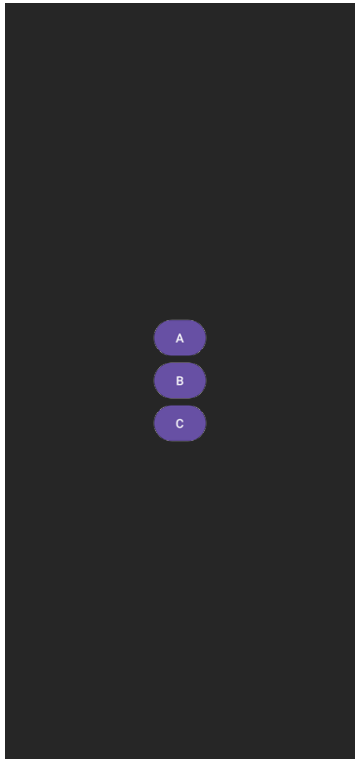



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val verticalChain = createVerticalChain(btA, btB, btC, chainStyle = ChainStyle.SpreadInside)
   constrain( verticalChain )
   {
      top.linkTo(parent.top, 50.dp)
      bottom.linkTo(parent.bottom, 50.dp)
   }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                   {
                     start.linkTo(parent.start); end.linkTo(parent.end)

                   }
       ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                   {
                     start.linkTo(btA.start)
                   }
       ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                   {
                     start.linkTo(btA.start)
                   }
       ) { Text("C") }
}
```

ConstraintLayout

Vertical Chain

packed

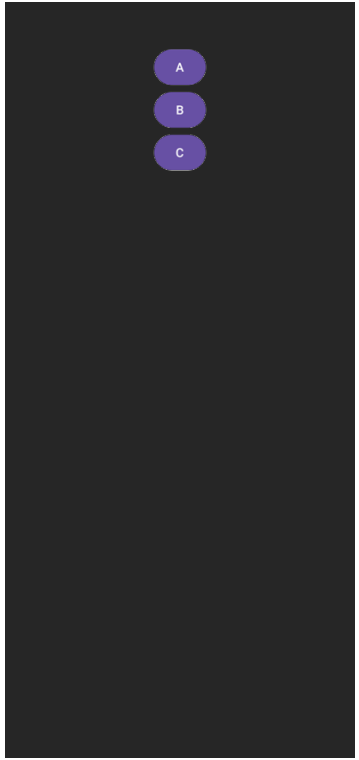



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val verticalChain = createVerticalChain(btA, btB, btC, chainStyle = ChainStyle.Packed)
   constrain( verticalChain )
   {
      top.linkTo(parent.top, 50.dp)
      bottom.linkTo(parent.bottom, 50.dp)
   }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                        {
                          start.linkTo(parent.start); end.linkTo(parent.end)

                        }
       ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                        {
                          start.linkTo(btA.start)
                        }
       ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                        {
                          start.linkTo(btA.start)
                        }
       ) { Text("C") }
}
```

ConstraintLayout

Vertical Chain

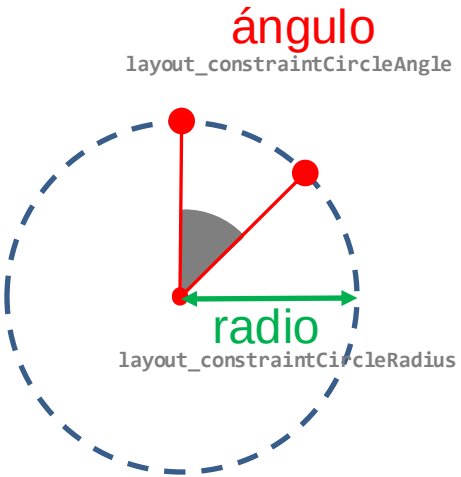
packed + bias

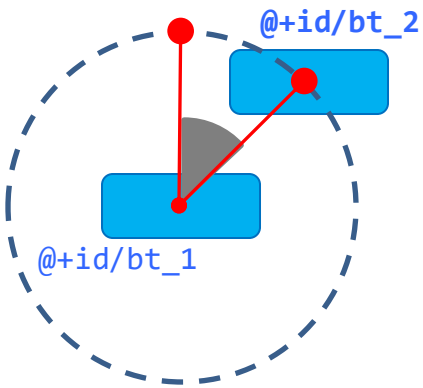



```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   val verticalChain = createVerticalChain(btA, btB, btC, chainStyle = ChainStyle.Packed)
   constrain( verticalChain )
   {
      top.linkTo(parent.top, 50.dp)
      bottom.linkTo(parent.bottom, 50.dp)
   }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btA)
                        {
                         start.linkTo(parent.start); end.linkTo(parent.end)
                         verticalBias = 0f
                        }
        ) { Text("A") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btB)
                        {
                         start.linkTo(btA.start)
                        }
        ) { Text("B") }
   Button(onClick = {},
        modifier = Modifier.constrainAs(btC)
                        {
                         start.linkTo(btA.start)
                        }
        ) { Text("C") }
}
```

ConstraintLayout

Align circle

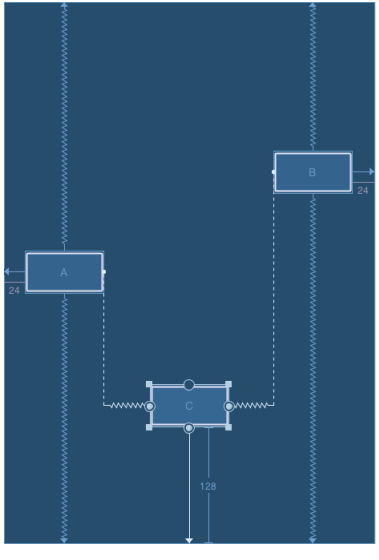




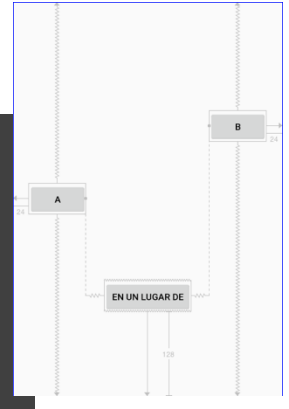

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
 val (btA, btB) = createRefs()
 Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                      {
                       centerTo(parent)
                      }
     ) { Text("Botón A") }

 Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                      {
                       circular(btA, 45f, 100.dp)
                      }
     ) { Text("Botón B") }
}
```

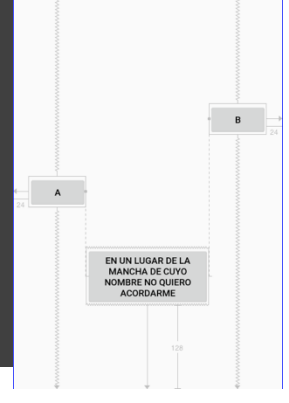
ConstraintLayout

Dimensions



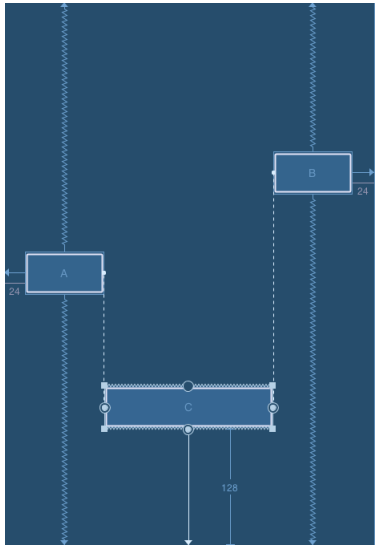

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                          centerVerticallyTo(other = parent, bias = 0.5f)
                          start.linkTo(anchor = parent.start, margin = 24.dp)
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                          centerVerticallyTo(other = parent, bias = 0.25f)
                          end.linkTo(anchor = parent.end, margin = 24.dp)
                        }
       ) { Text("Botón B") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btC)
                        {
                          start.linkTo(btA.end)
                          end.linkTo(btB.start)
                          bottom.linkTo(anchor = parent.bottom, margin = 128.dp)
                          width = Dimension.preferedWrapContent
                        }
       ) { Text("Botón C") }
}
```





ConstraintLayout

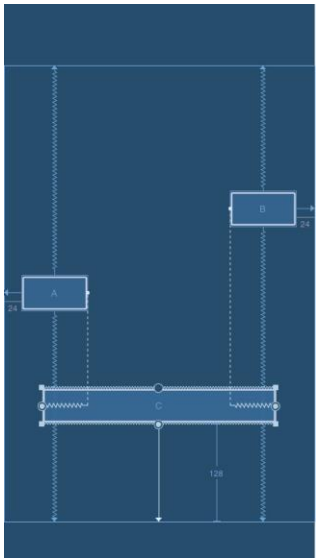
Dimensions




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                          centerVerticallyTo(other = parent, bias = 0.5f)
                          start.linkTo(anchor = parent.start, margin = 24.dp)
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                          centerVerticallyTo(other = parent, bias = 0.25f)
                          end.linkTo(anchor = parent.end, margin = 24.dp)
                        }
       ) { Text("Botón B") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btC)
                        {
                          start.linkTo(btA.end)
                          end.linkTo(btB.start)
                          bottom.linkTo(anchor = parent.bottom, margin = 128.dp)
                          width = Dimension.fillToConstraints
                        }
       ) { Text("Botón C") }
}
```

ConstraintLayout

Dimensions




```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{
   val (btA, btB, btC) = createRefs()
   Button(onClick = {}, modifier = Modifier.constrainAs(btA)
                        {
                          centerVerticallyTo(other = parent, bias = 0.5f)
                          start.linkTo(anchor = parent.start, margin = 24.dp)
                        }
       ) { Text("Botón A") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btB)
                        {
                          centerVerticallyTo(other = parent, bias = 0.25f)
                          end.linkTo(anchor = parent.end, margin = 24.dp)
                        }
       ) { Text("Botón B") }

   Button(onClick = {}, modifier = Modifier.constrainAs(btC)
                        {
                          start.linkTo(btA.end)
                          end.linkTo(btB.start)
                          bottom.linkTo(anchor = parent.bottom, margin = 128.dp)
                          width = Dimension.percent(0.7f)
                        }
       ) { Text("Botón C") }
}
```

ConstraintLayout

Dimensions · ratio

ConstraintLayout

Dimensions · ratio


```
Button(onClick = {}, modifier = Modifier.constrainAs(btRatio)
                    {
                       centerTo(parent)
                       height = Dimension.value(90.dp)
                       width = Dimension.ratio("w,16:9")
                    }
    ) { Text(”Ratio 16:9") }
```

ConstraintLayout

Dimensions · ratio


```
Button(onClick = {}, modifier = Modifier.constrainAs(btRatio)
                    {
                      centerTo(parent)
                      height = Dimension.ratio("h,16:9")
                      width = Dimension.value(160.dp)
                    }
    ) { Text("Ratio 16:9") }
```

ConstraintLayout

Guidelines

```
ConstraintLayout(modifier = Modifier.fillMaxSize())
{

    val verticalGuide20dpFromStart = createGuidelineFromStart(20.dp)
    val verticalGuide20percentFromStart = createGuidelineFromStart(0.2f)
    val verticalGuide20dpFromEnd = createGuidelineFromEnd(20.dp)
    val verticalGuide20percentFromEnd = createGuidelineFromEnd(0.2f)

    val horizontalGuide40dpFromTop = createGuidelineFromTop(40.dp)
    val horizontalGuide40percentFromTop = createGuidelineFromTop(0.4f)
    val horizontalGuide40dpFromBottom = createGuidelineFromBottom(40.dp)
    val horizontalGuide40percentFromBottom = createGuidelineFromBottom(0.4f)

    val bt = createRef()

    Button(onClick = {}, modifier = Modifier.constrainAs(bt)
                                {
                                  start.linkTo(verticalGuide20percent)
                                  end.linkTo(verticalGuide20percentFromEnd)

                                  top.linkTo(horizontalGuide40percentFromTop)
                                  bottom.linkTo(horizontalGuide40percentFromBottom)

                                  width = Dimension.fillToConstraints
                                  height = Dimension.fillToConstraints
                                }
}
```