САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет
по Лабораторная Работа №1
Дисциплина
«Проектирование мобильных приложений»

выполнил: Кривицкий В.В.
группа: 3530901/90202
преподаватель: Кузнецов А.Н.

Санкт-Петербург
2021

Репозиторий

https://github.com/OGSegu/AndroidLabs

1. Цели

- Изучить разработку в среде Android Studio
- Познакомиться с типичными составляющими андроид проекта
- Изучить возможности/свойства LinearLayout
- Изучить возможности/свойства ConstraintLayout

2. Задачи

LinearLayout

В соответствие с вариантом реализовать верстку следующих экранов при помощи LinearLayout

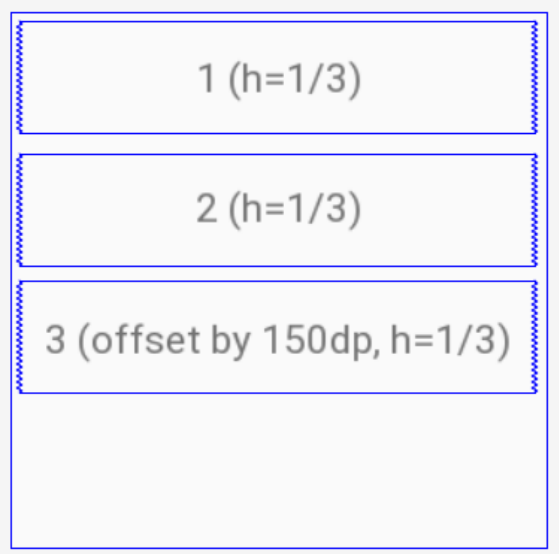

Рис. 1

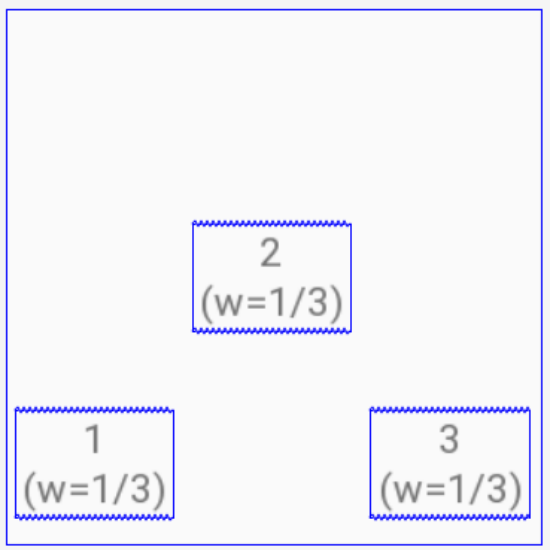

Рис. 2

ConstraintLayout

В соответствие с вариантом реализовать верстку следующи экранов при помощи ConstraintLayout

Рис. 1, Рис.2 – которые уже были реализованы при помощи LinearLayout

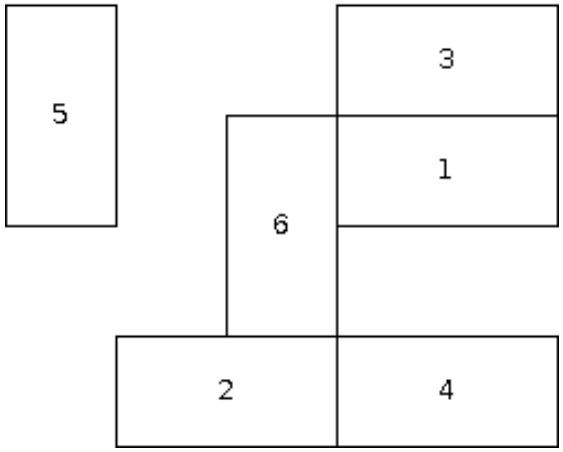

Рис. 3

3. Ход работы

Задача 1. LinearLayout

Реализуем верстку экрана с Рис.1
В папке с ресурсами создадим файл layout_1_12.xml, отметим его как LinearLayout, выставим android:orientation=”vertical”, так как работаем с верстской вертикальной ориентации экрана.
Создадим 3 виджета – AppCompatButton, TextView, ProgressBar. Для каждого из них выставим ширину по ширине Layout’a – match_parent. Явно укажем высоту элементов в 0dp, так как мы имплементируем гибкую реализацию по средствам задачи “веса” виджетов, который позволяет приоритезировать или уравновешивать занимаемое виджетом пространство на экране. Для наших целей указываем каждому виджету android:layout_weight = 1, так как их выоста должна быть 1/3 от размера экрана. Для самого нижнего элемента явно укажем отсуп снизу – android:layout_marginBottom = 150dp. Где константа указана в спецификации к заданию.

Код представлен в Листинге 1., а визуальный представление на Рис. 4

Листинг 1. Содержимое файла layout_1_12.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.appcompat.widget.AppCompatButton
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#FF00000" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/purple_200" />
    <ProgressBar
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginBottom="150dp"
        android:layout_weight="1"
        android:background="@color/teal_700" />
</LinearLayout>
```

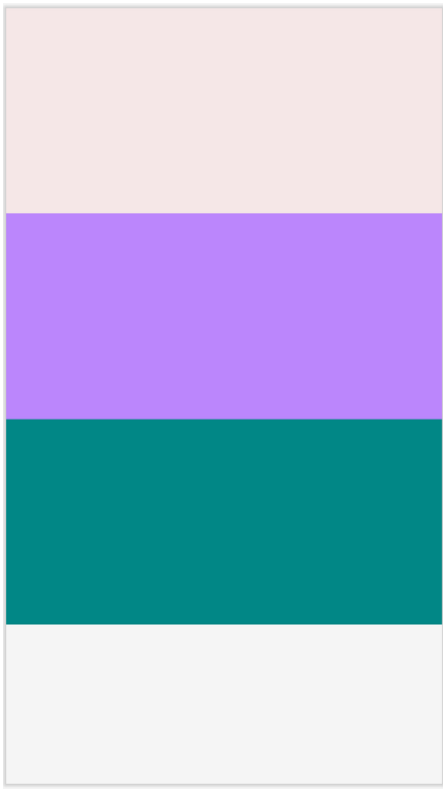

Рис. 4. Визуальное представление layout_1_12

Теперь реализуем аналогичный экран альтернативным вариантом. Предлагаю в качестве нижнего отступа воспользоваться виджетом “Space”, который по факту представляет из себя пустой элемент. По-моему мнению данное решение некоторым образом даже добавляет гибкости к нашей верстке, так как теперь наш отступ имеет множетсво свойств и состояний, а также напрямую не привязан к другим виджетам. Код представлен в Листинге 2

Листинг 2. Содержимое файла layout_1_12_alt.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.appcompat.widget.AppCompatButton
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#FF00000" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/purple_200" />
    <ProgressBar
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/teal_700" />
    <Space
        android:layout_width="match_parent"
        android:layout_height="150dp" />
</LinearLayout>
```

Визуальное представление аналогично Рис. 4

Теперь реализуем макет показанный на Рис.2
Идея данного макета очевидна, так по факту нам требуется сделать аналогичные манипуляции как и в первом задание, но расположить их горизонтально.
Выставим горизонтальную ориентацию экрана. Создадим те же 3 виджета, как и в задание выше, для каждого в качестве высоты укажем некоторое значение – например 50 dp, а ширину выставим в 0dp, так как мы хотим, что бы она менялась благодаря свойству width, которому мы выставляем значение – 1, потому что каждый элемент в ширину должен заниматься одинаковое пространство. Для реализации расположения элементов в разных частях экрана воспользуемся свойством layout_gravity и его говорящеми сами за себя значениями. Для первого и последнего элемента выставим значение – bottom, для центрального – center.
Код представлен в Листинге 3., а визуальное представление на Рис. 5

Листинг 3. Содержимое файла layout_1_21.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <androidx.appcompat.widget.AppCompatButton
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:layout_gravity="bottom"
        android:layout_weight="1"
        android:background="#FF00000" />
    <TextView
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:background="@color/purple_200" />
    <ProgressBar
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:layout_gravity="bottom"
        android:layout_weight="1"
        android:background="@color/teal_700" />
</LinearLayout>
```

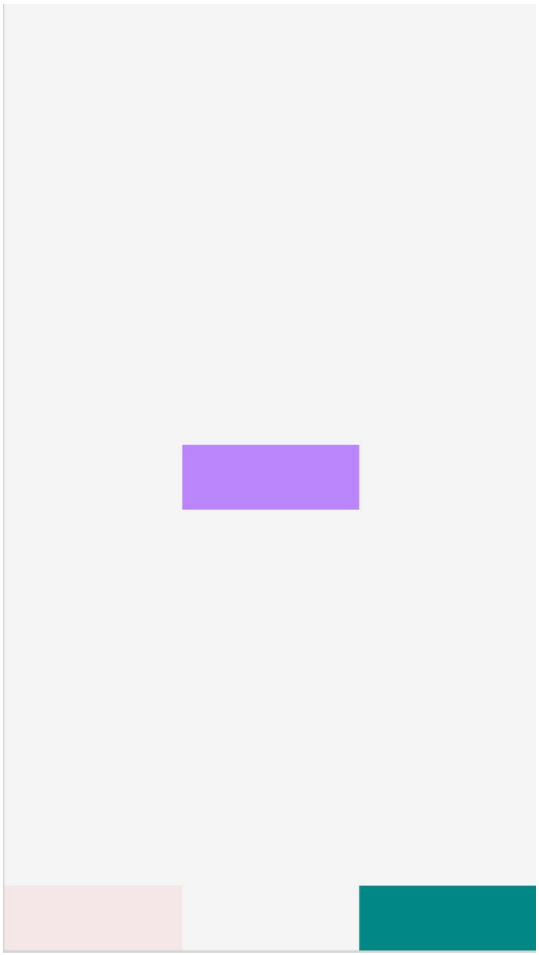

Рис. 5. Визуальное представление layout_1_21

Задача 2. ConstraintLayout

Данный макет позволяет нам гибко манипулировать виджетами, реализовано это, как минимум, благодаря большому кол-ву свойств “отношений” между объектами.

Реализуем экран с Рис.1 при помощи ConstraintLayout

Создадим 3 виджета – AppCompatButton, TextView, ProgressBar для каждого из низ укажем уникальный идентификатор по порядку их расположения сверху вниз: top, mid, bottom соответственно. Для упрощения описания будем использовать эти наименования. В качестве ширины каждого виджета укажем родительский элемент (на всю ширину экрана), а для высоты укажем 0dp, так как размер нашего виджета связан отношением других виджетов.

Выставим для top следующие свойства отношений:
layout_constraintTop_toTopOf = “parent” – это означет, что верхняя часть виджета позиционируется с верхней частью контейнера, в нашем случае ConstraintLayout.
layout_constraintBottom_toTopOf = “@id/mid”, это будет означать, то что нижняя часть текущего виджета позиционируется с верхней частью виджета mid.

Выставим для mid следующие свойства отношений:
layout_constraintBottom_toTopOf = “@id/bottom”.
layout_constraintTop_toBottomOf = “@id/top”, это будет означать, то что верхняя часть текущего виджета позиционируется с нижней частью виджета top.

Выставим для bottom следующие свойства отношений:
layout_constraintBottom_toTopOf = “@id/bottom”.
layout_constraintTop_toBottomOf = “@id/top”
layout_marginBottom = “150dp”, добавим внешний отступ снизу в соответсие с данной спецификацией.
Код представлен в Листинге 4. Визуальное представление на Рис. 4.

Листинг 4. Содержимое файла layout_2_12.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/top"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:background="#FF00000"
        app:layout_constraintBottom_toTopOf="@id/mid"
        app:layout_constraintTop_toTopOf="parent" />
    <TextView
        android:id="@+id/mid"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:background="@color/purple_200"
        app:layout_constraintBottom_toTopOf="@+id/bottom"
        app:layout_constraintTop_toBottomOf="@id/top" />
    <ProgressBar
        android:id="@+id/bottom"
        android:layout_width="match_parent"
```

```
        android:layout_height="0dp"
        android:layout_marginBottom="150dp"
        android:background="@color/teal_700"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@id/mid" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Реализуем экран с Рис. 2

Создадим 3 виджета – AppCompatButton, TextView, ProgressBar для каждого из низ укажем уникальный идентификатор по порядку их расположения слева направо: start, mid, end соответственно. Для упрощения описания будем использовать эти наименования. В качестве высоты каждого виджета укажем произвольное значение - 50dp, а для ширины укажем 0dp, так как размер нашего виджета связан отношением других виджетов.

Выставим для start следующие свойства отношений:
layout_constraintBottom_toBottomOf = “parent” – это будет означать, что нижняя часть виджета позиционируется с нижней частью контейнера, в нашем случае ConstraintLayout.
layout_constraintStart_toStartOf = “parent”, это будет означать, то что виджет начинается там, где начинается контейнер.
layout_constraintEnd_toStartOf = “@id/mid”, это будет означать, что начало виджета mid, будет позиционироваться с концом текущего виджета.

Выставим для mid следующие свойства отношений:
layout_constraintTop_toTopOf = “parent”, layout_constraintBottom_toBottomOf = “parent” – комбинация данных свойств отношений позволяет нам выравнивать элемент вертикально посередине.
layout_constraintEnd_toStartOf = “@id/end”, layout_constraintStart_toEndOf = “@id/start”.

Выставим для end следующие свойства отношений:
layout_constraintBottom_toBottomOf = “parent”.
layout_constraintEnd_toEndOf = “parent”
layout_constraintStart_toEndOf = “@id/mid”

Код представлен в Листинге 5. Визуальное представление на Рис. 5.

Листинг 5. Содержимое файла layout_2_21.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <androidx.appcompat.widget.AppCompatButton
        android:id="@+id/start"
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:background="#FF00000"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toStartOf="@id/mid"/>
    <TextView
        android:id="@+id/mid"
        android:layout_width="0dp"
        android:layout_height="50dp"
```

```
        android:background="@color/purple_200"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toStartOf="@id/end"
        app:layout_constraintStart_toEndOf="@id/start" />
    <ProgressBar
        android:id="@+id/end"
        android:layout_width="0dp"
        android:layout_height="50dp"
        android:background="@color/teal_700"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@id/mid"
        app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Задача 3. ConstraintLayout

Реализуем экран указанный на Рис. 3.

В данной задаче я буду осознанно не описывать свойства отношений, потому что большинство из них уже было упомянуто в предудыдущих заданиях. Опишу использованные новые и интересные для меня вещи.

Для упрощения написания отношений между виджетами, воспользуемся элементом – Guideline, который представляет из себя “линию помощи”, для которой или от которой мы можем описывать взаимоотношения. Всего было создано 2 Guideline один с вертикальной ориентацией, а другой с горизонтальной, их расположение регулируется свойством layout_constraintGuide_percent.

Много где используется свойство layout_constraintDimensionRatio, которое указывает пропорцию ширины на высоту. Например для элемента 2 указано значение 2, что означает, что ширина элемента в 2 раза больше ее высоты, таким образом во многих местах был реализован прямоугольник с “горизонтальной” ориентацией.

Также было использовано свойство layout_constraintHorizontal_bias = 1, для элемента 2. Так как для него указано отношение для начала и конца, то значение этого свойства притянет объект к элементу отношение, которого описано для конца.

Код представлен в Листинге 6.1 Визуальное представление на Рис. 6.

Листинг 6.1. Содержимое файла layout_3_12.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:background="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintDimensionRatio="1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent">
        <androidx.constraintlayout.widget.Guideline
             android:id="@+id/vertical_guideline"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:orientation="vertical"
             app:layout_constraintGuide_percent="0.6" />
        <androidx.constraintlayout.widget.Guideline
             android:id="@+id/top_guideline"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:orientation="horizontal"
             app:layout_constraintGuide_percent="0.2" />
        <TextView
             android:id="@+id/view1"
             style="@style/MyText"
             android:layout_width="0dp"
             android:layout_height="0dp"
             android:text="1"
             app:layout_constraintDimensionRatio="2"
             app:layout_constraintEnd_toEndOf="parent"
             app:layout_constraintStart_toStartOf="@id/vertical_guideline"
             app:layout_constraintTop_toBottomOf="@id/view3" />
        <TextView
             android:id="@+id/view2"
             style="@style/MyText"
             android:layout_width="0dp"
             android:layout_height="0dp"
             android:text="2"
             app:layout_constraintBottom_toBottomOf="parent"
             app:layout_constraintDimensionRatio="2"
             app:layout_constraintEnd_toEndOf="@id/vertical_guideline"
             app:layout_constraintHorizontal_bias="1"
             app:layout_constraintTop_toTopOf="@id/view4" />
        <TextView
             android:id="@+id/view3"
             style="@style/MyText"
             android:layout_width="0dp"
             android:layout_height="0dp"
             android:text="3"
             app:layout_constraintDimensionRatio="2"
             app:layout_constraintEnd_toEndOf="parent"
             app:layout_constraintStart_toStartOf="@id/vertical_guideline"
             app:layout_constraintTop_toTopOf="@id/top_guideline" />
        <TextView
             android:id="@+id/view4"
             style="@style/MyText"
             android:layout_width="0dp"
             android:layout_height="0dp"
             android:text="4"
             app:layout_constraintBottom_toBottomOf="parent"
             app:layout_constraintDimensionRatio="2"
             app:layout_constraintEnd_toEndOf="parent"
             app:layout_constraintStart_toStartOf="@id/vertical_guideline" />
        <TextView
             android:id="@+id/view5"
             style="@style/MyText"
             android:layout_width="0dp"
             android:layout_height="0dp"
             android:text="5"
             app:layout_constraintBottom_toBottomOf="@id/view1"
             app:layout_constraintEnd_toStartOf="@id/view2"
             app:layout_constraintStart_toStartOf="parent"
             app:layout_constraintTop_toTopOf="@id/top_guideline" />
```

```
        <TextView
            android:id="@+id/view6"
            style="@style/MyText"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:text="6"
            app:layout_constraintBottom_toTopOf="@id/view2"
            app:layout_constraintDimensionRatio="0.5"
            app:layout_constraintEnd_toEndOf="@id/vertical_guideline"
            app:layout_constraintTop_toTopOf="@id/view1" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

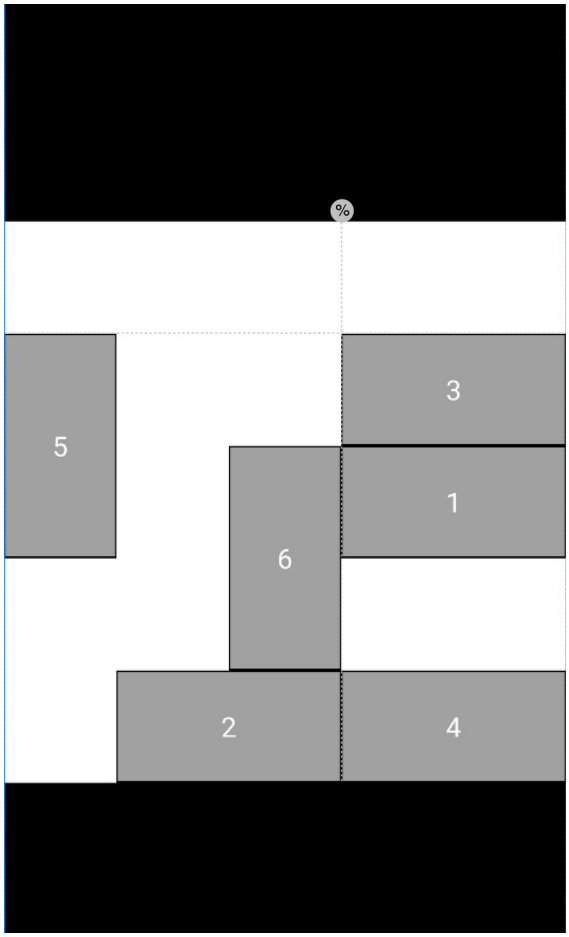

Рис 6. Визуальное представление layout_3_12

4. Вывод

В данной лабораторной работе были выполнены все поставленные цели и решены требуемые задачи.
Можно сделать вывод о том, когда стоит использовать LinearLayout, а когда ConstraintLayout. Для самых примитивных экранов, где элементы расположены друг за другом быстрее и предпочтительне будет использовать LinearLayout, в остальных случаев, где требуется задавать более сложные свойства и отношения между объектами, прекрасно справляется ConstraintLayout.