Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерный наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий.

**Отчёт по лабораторной работе №1**

**Дисциплина**: Проектирование мобильных приложений

**Тема**: Layouts. Вариант 13

Выполнил студент гр. 3530901/80201: ­­­­­­­­­­­­­ ­ В.Д. Петров

(подпись)

Преподаватель: ­А.Н. Кузнецов

(подпись)

Санкт-Петербург

2020

**СОДЕРЖАНИЕ**

Введение…………………………………………………………………………….3

Задача №1. Linear layout……………………………………………………………4

Задача №2. Constraint layout………………………………………………………12

Задача №3. Constraint layout……………………………………………………....16

Заключение………………………………………………………………………..20

Список использованных источников…………………………………………….21

**ВВЕДЕНИЕ**

Layout ресурсы являются фундаментом для большинства приложений, это некий визуальный шаблон пользовательского интерфейса приложения, который позволяет управлять виджетами, их свойствами и расположением. Целью данной работы является ознакомление со средой разработки Android Studio, изучение основных принципов верстки layout с использованием View и ViewGroup, а также изучение основных возможностей и свойств Linear layout и Constraint layout.

**ЗАДАЧА №1. LINEAR LAYOUT**

LinearLayout это ViewGroup, которая располагает все элементы приложения в одном направлении – вертикальном или горизонтальном. Направление можно поменять с помощью атрибута android:orientation.

С помощью атрибута android:layout\_weight можно назначать веса отдельным дочерним элементам. Этот атрибут присваивает элементу “важность” с точки зрения того, сколько места он должен занимать на экране. Большее значение веса позволяет ему расширяться и заполнять все оставшееся пространство. Дочерние элементы указывают значение веса, и затем все свободное место в ViewGroup распределяется между ними в пропорциях указанного веса. По умолчанию вес элемента равен нулю.

Атрибут android:gravity определяет как LinearLayout будет располагать элементы внутри себя. Атрибут android:linear\_gravity определяет как данный элемент будет располагаться в своей группе, может принимать значения bottom, center, center\_horizontal, center\_vertical, clip\_horizontal, clip\_vertical, end, fill, fill\_horizontal, fill\_vertical, left, right, start, top.

Атрибут android:layout\_height определяет базовую высоту для View, может принимать значения match\_parent, заполняя все пространство родительского View, или же wrap\_content, заполняя столько пространства, сколько необходимо для контента данного элемента.

По заданию 1 необходимо создать layout ресурсы для макетов экрана (Рис.1.1 и Рис.1.2) с использованием LinearLayout. В рамках каждого макета необходимо использовать разные виджеты для разных элементов.

Исходный код для задания 1 и последующих находится на github по ссылке:

<https://github.com/ADsty/labs-android_labs/tree/master/lab1/src/main/res/layout>

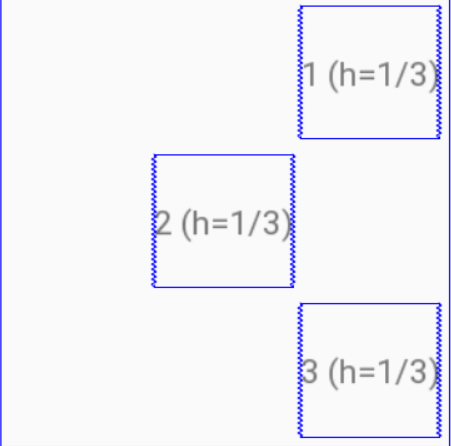


Рис.1.1 Изображение №1 согласно варианту

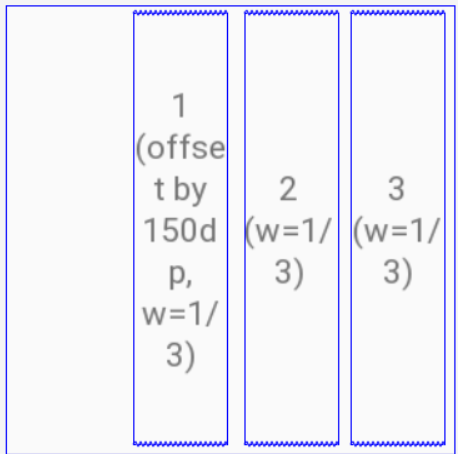


Рис.1.2 Изображение №2 согласно варианту

Для изображения №1 layout ресурс строился с использованием вертикального LinearLayout, android:gravity=“end” атрибута, который все объекты внутри LinearLayout отправляет в конец контейнера, а также использования android:layout\_gravity=”center” для центрального виджета, чтобы отправить виджет в центр контейнера. Всем виджетам был присвоен одинаковый вес с помощью android:layout\_weight=”1” атрибута у каждого виджета, а также android:layout\_height="match\_parent”, в результате виджеты по вертикали поделили между собой экран на три равные части. По горизонтали виджеты используют столько пространства, сколько им нужно, с помощью атрибута android:layout\_width=”wrap\_content”. В результате получается следующий layout ресурс (Рис.1.3)

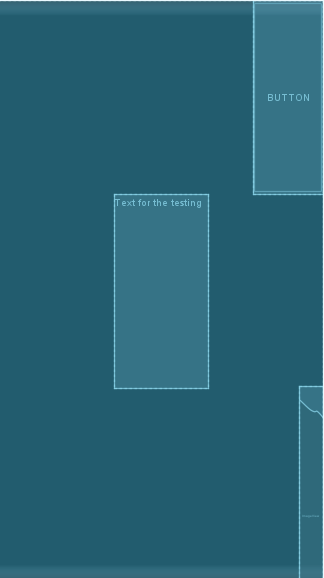


Рис.1.3 layout ресурс для изображения №1

Поскольку различные виджеты используют различное пространство для отображения, то для большей демонстрации корректности работы были использованы одинаковые виджеты (Рис.1.4)

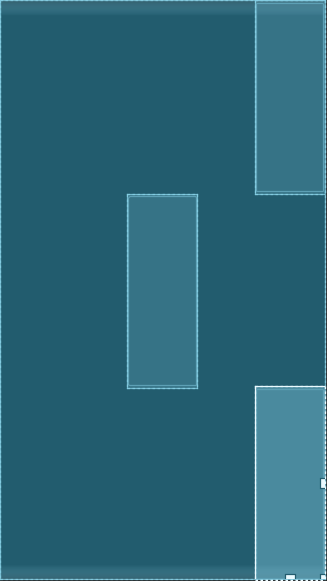


Рис.1.4 layout ресурс с одинаковыми виджетами

Получившийся XML код приведен ниже (Листинг 1)

Листинг 1 – код решения для 1 изображения с использованием LinearLayout

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout 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:gravity="end"  
 android:orientation="vertical">  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="Button" />  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="center"  
 android:layout\_weight="1"  
 android:text="Text for the testing" />  
  
 <ImageView  
 android:id="@+id/imageView4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 app:srcCompat="@android:drawable/sym\_action\_email" />  
  
</LinearLayout>

Данный код не содержит излишних атрибутов, поскольку удаление любого из озвученных выше атрибутов приведет к некорректной верстке, например при удалении любого из android:layout\_weight=”1” приведет к сжатию всех виджетов с весом 1, т.к. атрибут с весом 0 заполнит все свободное по высоте пространство (Рис.1.5)

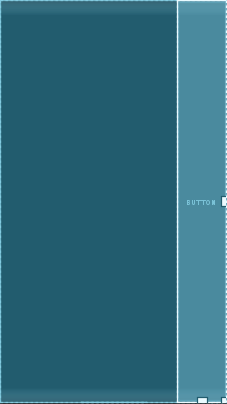


Рис.1.5 Пример некорректной верстки

Для изображения №2 была использована горизонтальная ориентация LinearLayout, а также атрибут android:layout\_marginLeft=”150dp” для создания отступа слева на 150dp, у всех виджетов был выставлен вес 1, а также height и width в значение “match\_parent” для заполнения экрана по вертикали, а также деления оставшейся после отступа части экрана на три равные части по горизонтали. В результате получился следующий layout ресурс (Рис.1.6)

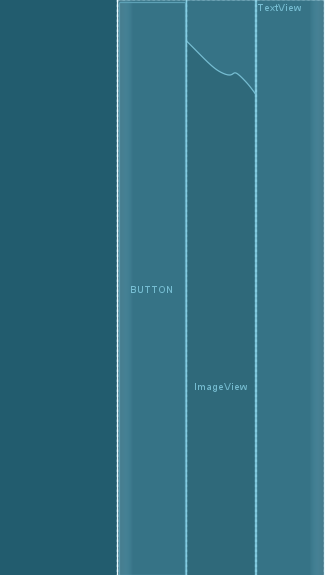


Рис.1.6 layout ресурс для изображения №2

Получившийся XML код приведен ниже (Листинг 2)

Листинг 2 – код решения для 2 изображения с использованием Linear layout

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout 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:layout\_marginLeft="150dp"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="Button" />  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 app:srcCompat="?android:attr/textCheckMark" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="TextView" />  
</LinearLayout>

Данный код не содержит излишних атрибутов, т.к. удаление любого из атрибутов приведет к некорректной верстке, например при удалении атрибута android:layout\_marginLeft=”150dp” пропадет отступ слева (Рис.1.7)

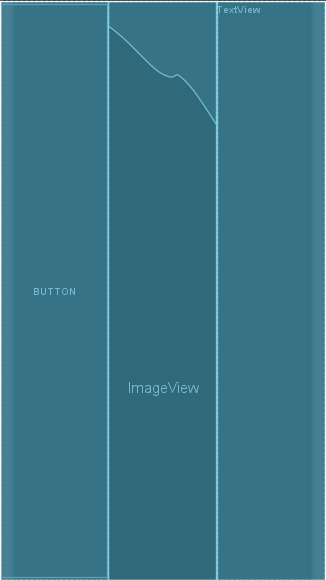


Рис.1.7 Пример некорректной верстки

Другой вариант решения данной задачи – использование атрибута android:paddingLeft=”150dp”, который в отличие от атрибута android:marginLeft перемещает виджеты не снаружи, а внутри себя.

Полученный XML код данного ресурса представлен ниже (Листинг 3)

Листинг 3 – код альтернативного решения для 2 изображения

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout 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="horizontal"  
 android:paddingLeft="150dp">  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="Button" />  
  
 <ImageView  
 android:id="@+id/imageView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 app:srcCompat="@android:drawable/checkbox\_on\_background" />  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="TextView" />  
</LinearLayout>

**ЗАДАЧА №2. CONSTRAINT LAYOUT**

ConstraintLayout позволяет создавать большие и комплексные layout ресурсы с простой иерархией (без вложенных ViewGroup). Чтобы определить позицию виджета в ConstraintLayout необходимо как минимум одно вертикальное и одно горизонтальное соединение для данного виджета. Соединения показывают взаимосвязь виджета с другими view, родительским layout ресурсом. С помощью атрибута android:layout\_constraintDimensionRatio можно указать соотношение сторон в виде ширина:высота, для этого одна из сторон должна быть указана “0dp” или “match\_constraints”. Атрибуты android:layout\_constraintHorizontal\_weight и android:layout\_constraintVertical\_weight работают точно так же, как и весы в LinearLayout. Необходимо повторить изображения №1 и №2 с использованием ConstraintLayout (Рис.2.1 и Рис.2.3).

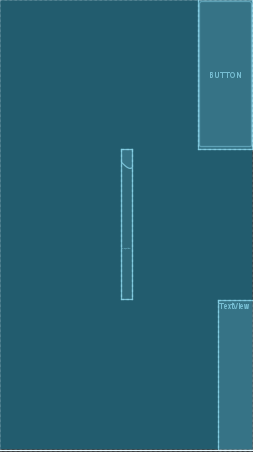


Рис.2.1 Изображение №1 с использованием ConstraintLayout

Получившийся XML код представлен ниже (Листинг 4)

Листинг 4 – код решения для 1 изображения с использованием Constraint layout

<?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">  
  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:text="Button"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@id/imageView3"/>  
  
 <ImageView  
 android:id="@+id/imageView3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 app:layout\_constraintTop\_toBottomOf="@id/button4"  
 app:layout\_constraintBottom\_toTopOf="@+id/textView4"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:srcCompat="@android:drawable/checkbox\_on\_background" />  
  
 <TextView  
 android:id="@+id/textView4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="0dp"  
 android:text="TextView"  
 app:layout\_constraintTop\_toBottomOf="@id/imageView3"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent" />  
</androidx.constraintlayout.widget.ConstraintLayout>

Данный код не содержит избыточных атрибутов, тк удаление любого атрибута приведет к неправильной верстке. Например, удаление одного из соединений приведет к следующему результату (Рис 2.2)

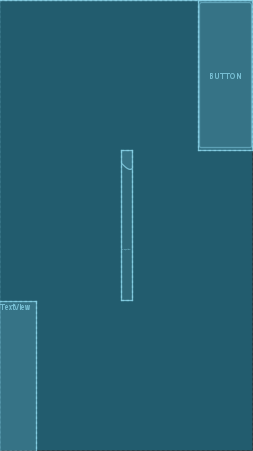


Рис.2.2 Пример неправильной верстки

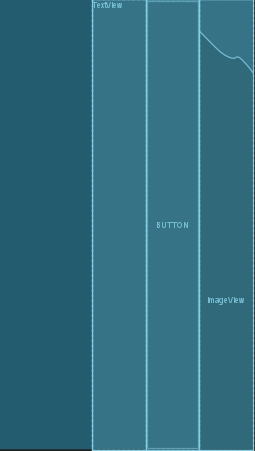


Рис.2.3 Изображение № 2 с использованием ConstraintLayout

Полученный XML код представлен ниже (Листинг 5)

Листинг 5 – код решения для 2 изображения с использованием Constraint layout

<?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:layout\_marginLeft="150dp">  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:text="TextView"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/button3"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:text="Button"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/imageView2"  
 app:layout\_constraintStart\_toEndOf="@+id/textView3"  
 app:layout\_constraintTop\_toTopOf="parent"/>  
  
 <ImageView  
 android:id="@+id/imageView2"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/button3"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@android:drawable/checkbox\_on\_background" />  
</androidx.constraintlayout.widget.ConstraintLayout>

Данный код не содержит избыточных атрибутов, тк. удаление любого из них приведет к неправильной верстке (Рис.2.4)

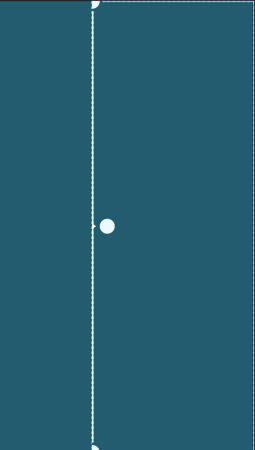


Рис.2.4 Пример неправильной верстки

**ЗАДАЧА №3. CONSTRAINT LAYOUT**

В данном задании необходимо построить изображение №3 (Рис.3.1) с использованием ConstraintLayout. Данное изображение было получено следующим образом: для всех виджетов были выставлены значения длины и ширины 0dp, а также соотношения сторон 1:2 или 2:1, в зависимости от расположения виджета – горизонтальное или вертикальное. Затем были соединены виджеты 5, 2 и 6, для виджетов 5 и 6 был выставлен вес 1, а для второго виджета вес 2, после чего были созданы оставшиеся виджеты и соединены между собой. В результате получилось следующее изображение (Рис.3.2)

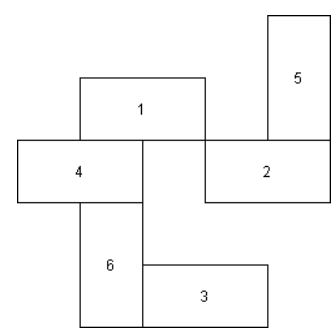


Рис.3.1 Изображение №3 согласно варианту

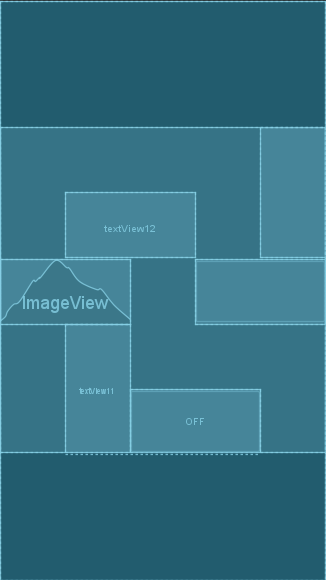


Рис.3.2 Изображение №3 с использованием ConstraintLayout

Полученный XML код приведен ниже (Листинг 6)

Листинг 6 – код решения для изображения 3

<?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">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintDimensionRatio="1:1"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <TextView  
 android:id="@+id/textView8"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintDimensionRatio="1:2"  
 app:layout\_constraintVertical\_weight="2"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@+id/textView9" />  
  
 <Button  
 android:id="@+id/textView9"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintVertical\_weight="1"  
 app:layout\_constraintDimensionRatio="2:1"  
 app:layout\_constraintBottom\_toTopOf="@+id/textView11"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView8" />  
  
 <ImageView  
 android:id="@+id/textView10"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintDimensionRatio="2:1"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView8"  
 app:layout\_constraintBottom\_toBottomOf="@id/textView9"/>  
  
 <ImageButton  
 android:id="@+id/textView11"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintDimensionRatio="1:2"  
 app:layout\_constraintVertical\_weight="2"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView10"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView9" />

<EditText  
 android:id="@+id/textView12"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintDimensionRatio="2:1"  
 app:layout\_constraintBottom\_toBottomOf="@+id/textView8"  
 app:layout\_constraintEnd\_toStartOf="@+id/textView9"  
 app:layout\_constraintStart\_toStartOf="@+id/textView11" />  
  
 <ToggleButton  
 android:id="@+id/textView13"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintDimensionRatio="2:1"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/textView8"  
 app:layout\_constraintStart\_toEndOf="@+id/textView11" />  
  
  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

Данный код не содержит избыточных атрибутов, ибо удаление любого из них приведет к неправильной верстке (Рис.3.3)

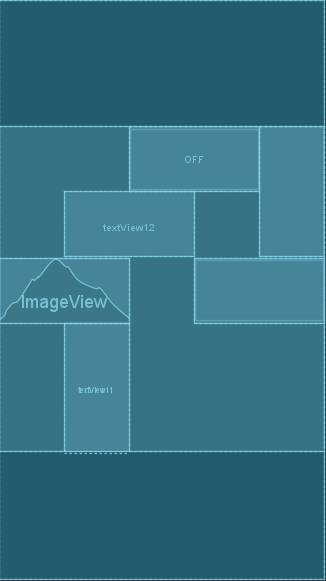


Рис.3.3 Пример неправильной верстки

**ЗАКЛЮЧЕНИЕ**

В данной работе были изучены основные принципы верстки layout ресурсов с использованием LinearLayout и ConstraintLayout. С использованием LinearLayout были построены вертикальная и горизонтальные компоновки различных виджетов с применением атрибутов layout\_weight, gravity и других. С использованием ConstraintLayout были построены аналогичные изображения, а также сложное изображение с использованием атрибута layout\_constraintDimensionRatio, было изучено что весы для ConstraintLayout работают аналогично весам для LinearLayout. С помощью ConstraintLayout можно построить любую компоновку виджетов на экране, однако она подходит для того, чтобы компоновать некоторое число View и ViewGroup в сложный layout ресурс, который иначе требовал бы сложной иерархии ViewGroup. Для более простых дизайнов удобнее будет использовать LinearLayout.

Исходный код для лабораторных работ находится на [github](https://github.com/ADsty/labs-android_labs) (<https://github.com/ADsty/labs-android_labs> )

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

<https://classroom.udacity.com/courses/ud9012>

<https://developer.android.com/guide/topics/ui/layout/linear>

<https://developer.android.com/training/constraint-layout>