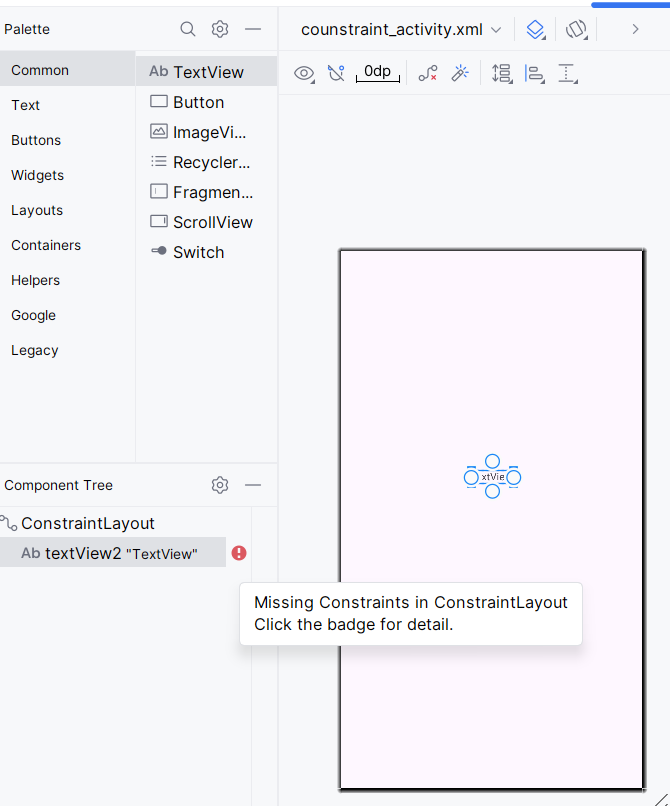
**ConstraintLayout**

ConstraintLayout очень похож на RelativeLayout, потому что элементы располагаются в соответствии между элементами и родительским макетом, но он намного более гибкий.

#### Опорные точки

Предположим, вы перетаскиваете элемент TextView в визуальном редакторе ConstraintLayout в Android Studio. Сразу после перетаскивания вы заметите ошибку. Это просто означает, что созданный элемент не ограничен. Если не исправить это, то представление не будет отображаться должным образом при запуске в приложении.



#### Вы также можете наведя курсор на элемент увидеть различные точки, которые можно назвать **маркерами или опорными точками**.

Нажмите на любую точку и перетащите ее, чтобы установить связь с чем-то еще вокруг.

Нужно будет выполнить как минимум два соединения точек с чем-то другим, чтобы установить расположение элемента в контейнере.

Относительное позиционирование в ConstraintLayout:

Относительное позиционирование является наиболее важной особенностью ConstraintLayout и основа построения элементов внутри контейнера. Различные варианты ограничений, которые он предлагает, работают по отношению друг к другу. Относительное позиционирование работает только по вертикальной и горизонтальной осям.

Ниже приведены некоторые ограничения, которые можно использовать для установки положения относительно другого элемента:

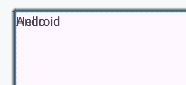
* layout\_constraintLeft\_toLeftOf : левая граница элемента располагается относительно левой границы другого элемента
* layout\_constraintLeft\_toRightOf : левая граница элемента расположена относительно правой границы другого элемента
* layout\_constraintRight\_toLeftOf: правая граница элемента расположена относительно левой границы другого элемента
* layout\_constraintRight\_toRightOf: правая граница элемента расположена относительно правой границы другого элемента.

**FrameLayout**

FrameLayout - это один из самых простых контейнеров для организации элементов. Элементы на этом контейнере располагаются друг на друге. Их можно переместить друг от друга изменив отступы либо с помощью гравитации. Этот контейнер предназначен для блокирования одного объекта другим или создания сложных многослойный объектов, не рекомендуется использовать его в других ситуациях, потому что может быть сложно отображать отдельные элементы в определенной области экрана без наложения друг на друга.

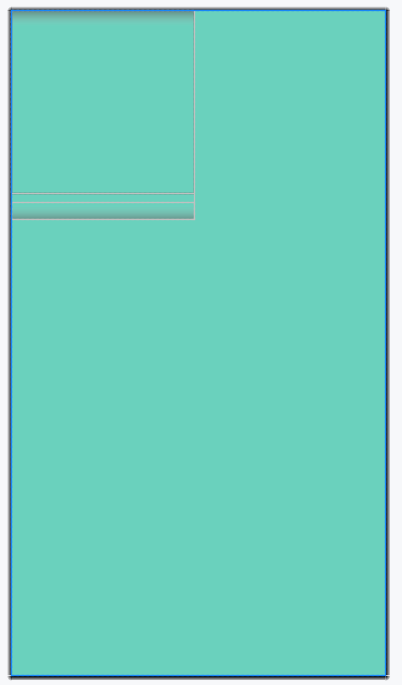
Вы можете добавить несколько дочерних элементов в FrameLayout и управлять их положением, назначая каждому дочернему элементу атрибут android:layout\_gravity.





Атрибут Foreground определяет цвет или элемент который рисуется поверх содержимого.





Благодаря Foreground добавленные элементы отображаются но скрываются за цветным фоном.

Атрибут Visibility определяет видим, невидим или скрыт элемент.

* android:visibility="visible" - элемент виден
* android:visibility="invisible" - элемент не виден, но располагается на макете и влияет на расположение остальных элементов
* android:visibility="gone"- элемент отсутствует, он не виден, и не влияет на расположение остальных элементов

TableLayout

В Android TableLayout используется для упорядочивания группы представлений по строкам и столбцам. В TableLayout не отображается линия границы для столбцов, строк или ячеек.



Для построения строки в таблице мы будем использовать элемент TableRow. TableRow - это дочерние элементы TableLayout. Каждый TableRow содержит в себе ячейки - элементы которые находятся в строчке.



Атрибут stretchColumns используется для изменения ширины столбца, которая устанавливается равной ширине самого широкого столбца, но мы также можем растянуть столбцы, чтобы занять доступное свободное пространство, используя этот атрибут. Значением, которое присваивается этому атрибуту, может быть номером одного столбца или список номеров столбцов, разделенных запятыми (1, 2, 3 ... n).

Если значение равно 0, то растягивается первый столбец, если значение равно 1, то растягивается второй столбец, чтобы занять любое доступное пространство в строке, потому что номера столбцов начинаются с 0.

Если значение равно 0,1, то и первый, и второй столбцы таблицы растягиваются, чтобы занять доступное пространство в строке.

Если значение равно ‘\*’, то все столбцы растягиваются, чтобы занять доступное пространство.



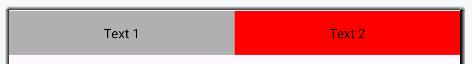
При android:stretchColumns="0"



При android:stretchColumns="1"



При android:stretchColumns="\*"



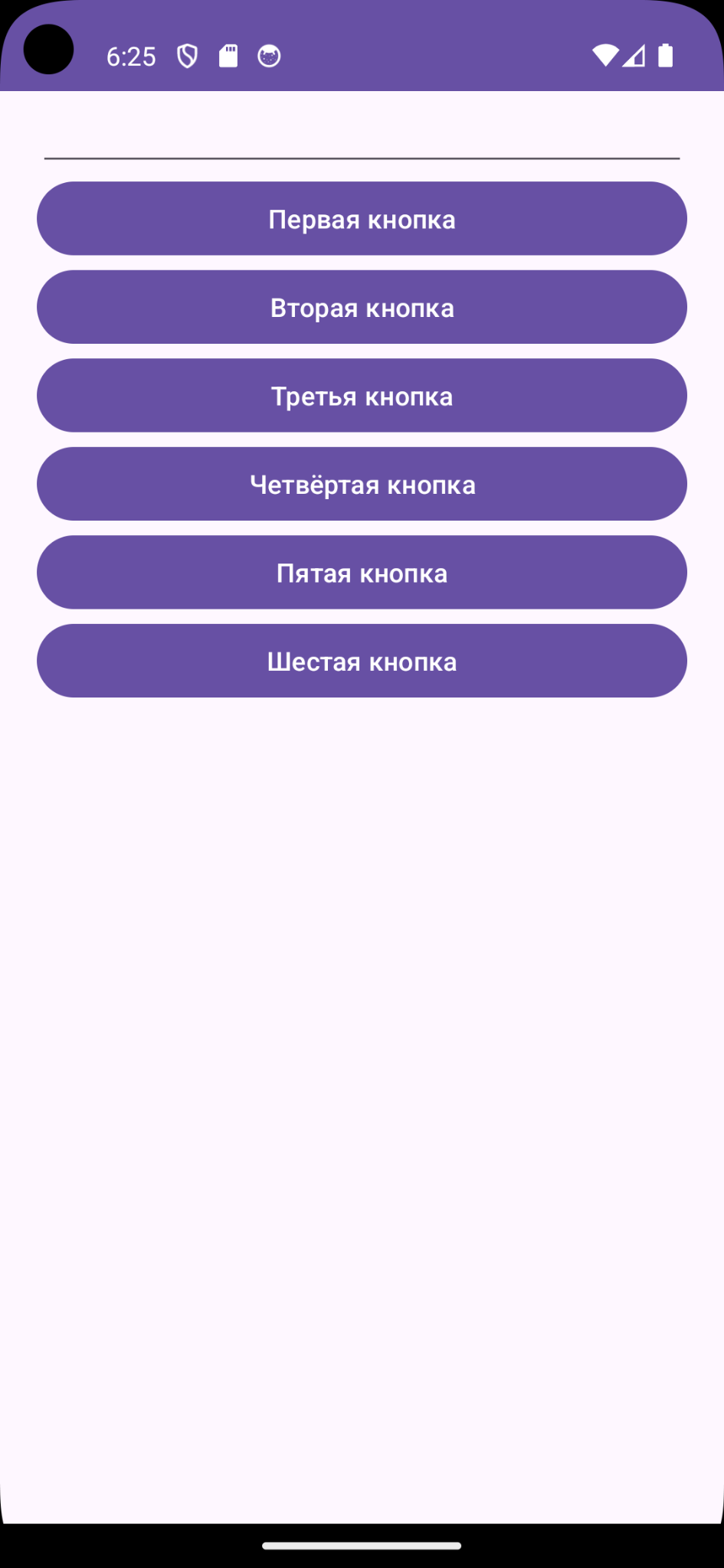
Атрибут collapseColumns используется для сворачивания или невидимости столбцов макета таблицы. Эти столбцы являются частью таблицы, но невидимы. Если значение равно 0, то первый столбец выглядит свернутым.

**Настройка альбомного макета**

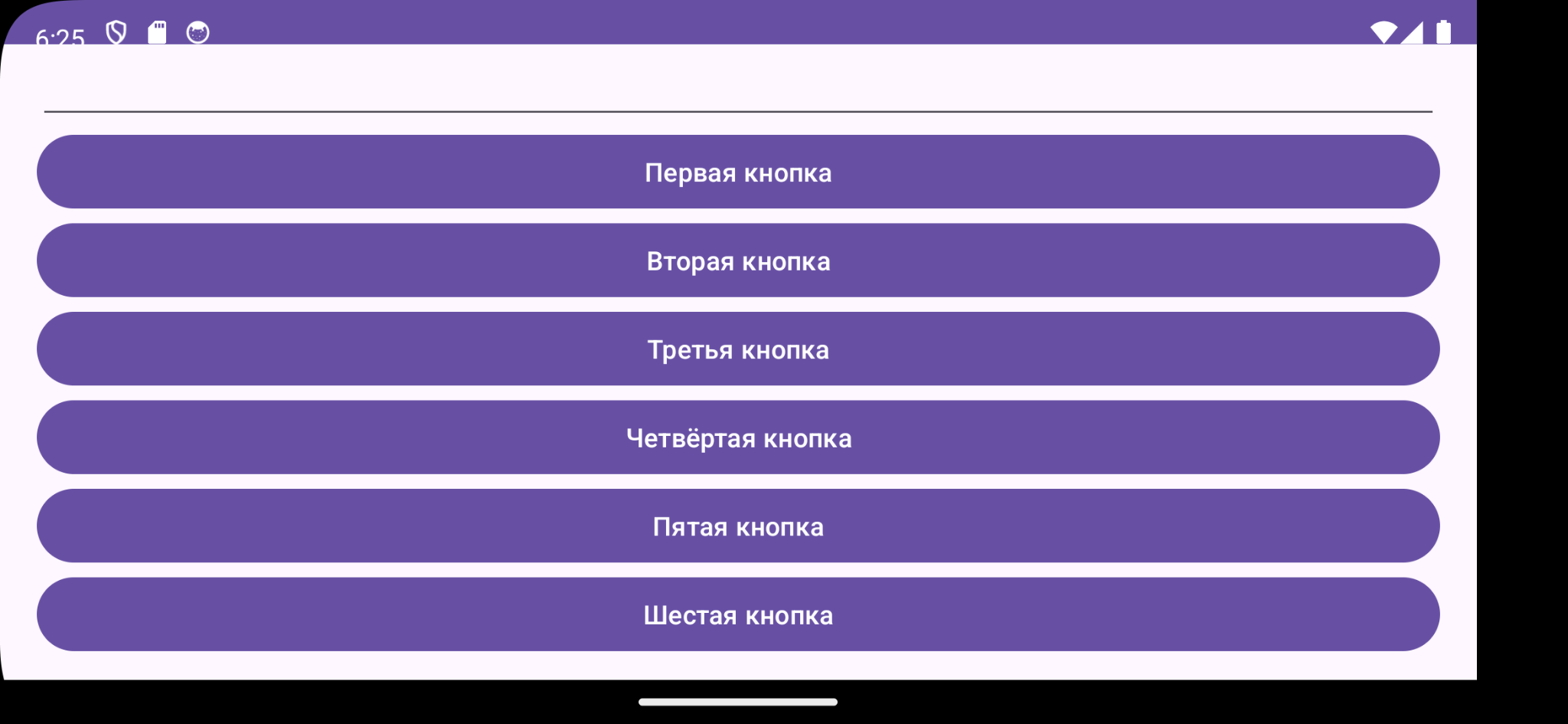
Первые мобильные устройства не умели менять ориентацию экрана. Необходимо было самостоятельно создавать программы, которые умели переключаться в разные режимы, только с появлением смартфонов эту возможность стали обязательно включать в настройки устройства.

Всего существует два режима - портретный и альбомный. По умолчанию на большинстве телефонов используется портретный режим, альбомный режим приоритезирован для планшетов и мониторов.

Предположим, у вас в приложении имеется одно текстовое поле и шесть кнопок.

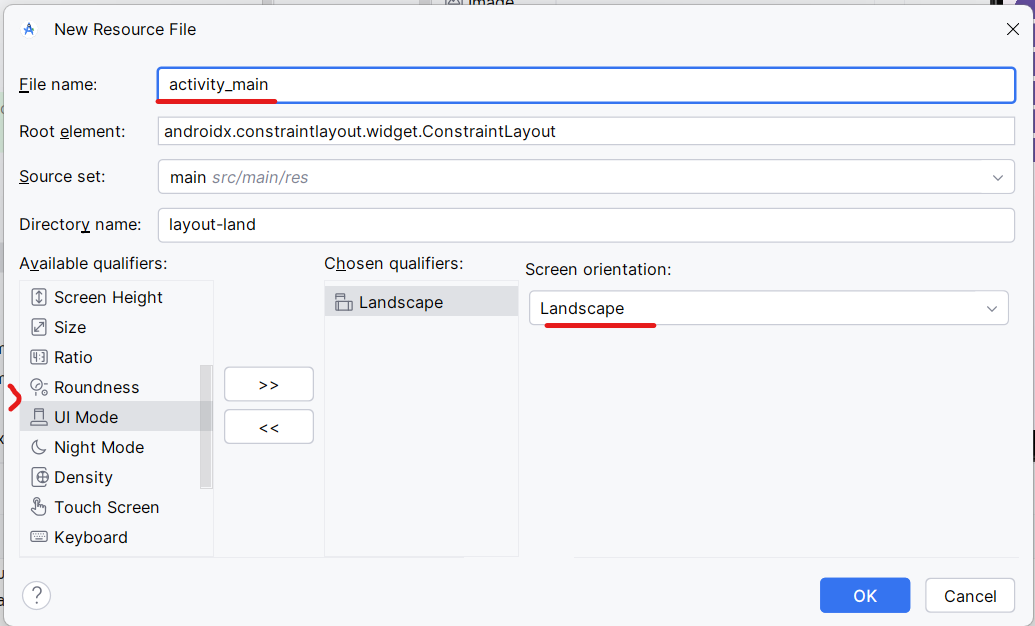


Если попробовать повернуть эмулятор на 90 градусов то либо обнаружатся проблемы с интерфейсом либо приложение не будет менять сторону экрана.

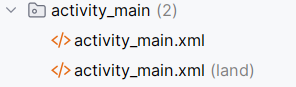


Чтобы избежать таких проблем, необходимо по-другому скомпоновать интерфейс. Например, в данном случае можно расположить их не подряд друг за другом, а разбить на пары, с помощью контейнера TableLayout.

Чтобы создать дополнительную версию вашего макета с другой ориентацией экрана нажмите правой кнопкой мыши на папку layout и выберете New -> Android Resource File. В открывшимся окне в списке Available qulifers выберите Orientation и правее в появившемся окне Screen orientation выберите Landscape. Присвойте имя файла такое же как и файл для которого вы делаете дополнительную ориентацию экрана.



Если файл был создан успешно то Android Studio определит их в одну папку. Один из файлов будет отмечен как land



Откройте созданный файл и установите следующий дизайн интерфейса.



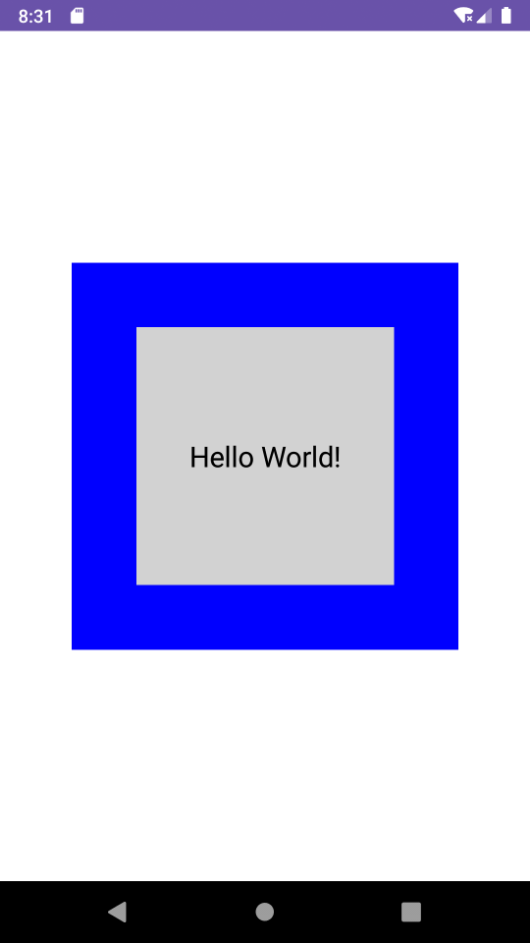
И так далее пока не получится следующий макет



Теперь при запуске приложения и повороте на 90 градусов вместо созданного нами первого экрана открывается альбомная версия.

**Задание для самостоятельной работы:**

1. Разработайте приложение с интерфейсом соответствующим следующему макету (используя FrameLayout)



1. Разработайте приложение с интерфейсом соответствующим следующему макету (используя TableLayout)

