**Андроид-приложение с вкладками – TabLayout с ViewPager2 на Kotlin**

**Делаем на основании приложения из урока 18!!**

**Добавление TabLayout в макет разметки экрана**

Чтобы добавить вкладки на экран, нужно открыть макет разметки и добавить компонент com.google.android.material.tabs.TabLayout в верхней части экрана:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <?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"      xmlns:tools="http://schemas.android.com/tools"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      tools:context=".MainActivity">        <com.google.android.material.tabs.TabLayout          android:id="@+id/tab\_layout"          app:tabMode="scrollable"          app:tabIndicatorColor="@color/teal\_200"          app:tabIndicatorHeight="4dp"          android:layout\_width="match\_parent"          android:layout\_height="wrap\_content"          app:layout\_constraintEnd\_toEndOf="parent"          app:layout\_constraintStart\_toStartOf="parent"          app:layout\_constraintTop\_toTopOf="parent" />        <androidx.viewpager2.widget.ViewPager2          android:id="@+id/pager"          android:layout\_width="match\_parent"          android:layout\_height="0dp"          app:layout\_constraintBottom\_toBottomOf="parent"          app:layout\_constraintLeft\_toLeftOf="parent"          app:layout\_constraintRight\_toRightOf="parent"          app:layout\_constraintTop\_toBottomOf="@+id/tab\_layout"/>    </androidx.constraintlayout.widget.ConstraintLayout> |

Для корректного размещения нужно изменить компонент ViewPager2 – высоту укажем 0dp. Таким образом, высоту компонента будет регулировать корневой ConstraintLayout по заданным ограничениям. А вместо ограничения app:layout\_constraintTop\_toTopOf=»parent» поставим app:layout\_constraintTop\_toBottomOf=»@+id/tab\_layout» – чтобы верх компонента ViewPager2 был ограничен не верхней границей родительского компонента, а нижней границей компонента TabLayout.

Рассмотрим подробнее компонент com.google.android.material.tabs.TabLayout. Свойство app:tabMode=»scrollable» обеспечивает размещение в видимой части экрана только нескольких вкладок, остальные будут доступны в процессе прокрутки. Если мы не укажем это свойство, то в видимой части экрана будут одновременно отображаться все вкладки, и при большом их количестве визуальное восприятие будет затруднено.

Свойство app:tabIndicatorColor=»@color/teal\_200″ указывает цвет, а app:tabIndicatorHeight=»4dp» – толщину индикатора вкладки.

Далее идут свойства ширины – указываем по родителю – и высоты – указываем по содержимому.

Последние три свойства – ограничения верхней части и боковых сторон компонента по родителю.

**Реализация вкладок в MainActivity**

Открываем класс MainActivity и пишем реализацию вкладок:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package info.fandroid.viewpager2app    import android.os.Bundle  import androidx.fragment.app.FragmentActivity  import androidx.viewpager2.widget.ViewPager2  import com.google.android.material.tabs.TabLayout  import com.google.android.material.tabs.TabLayoutMediator    class MainActivity : FragmentActivity() {        private lateinit var adapter: NumberAdapter      private lateinit var viewPager: ViewPager2      private lateinit var tabLayout: TabLayout        override fun onCreate(savedInstanceState: Bundle?) {          super.onCreate(savedInstanceState)          setContentView(R.layout.activity\_main)            adapter = NumberAdapter(this)          viewPager = findViewById(R.id.pager)          viewPager.adapter = adapter            tabLayout = findViewById(R.id.tab\_layout)          TabLayoutMediator(tabLayout, viewPager) { tab, position ->              tab.text = "TAB ${(position + 1)}"          }.attach()      }    } |

Инициализируем TabLayout так же, как мы это делали с ViewPager2 на прошлом уроке. Сначала объявляем переменную с ленивой инициализацией. В onCreate находим компонент TabLayout по идентификатору в макете разметки и связываем его с переменной.

Для синхронизации компонента TabLayout с ViewPager2, установки текста заголовков вкладок, а также стиля вкладок, используется класс TabLayoutMediator. Судя из его названия, это посредник, который выполняет связывание и согласование позиций списка вкладок со списком страниц ViewPager2, он слушает коллбек ViewPager2 и в соответствии с прокруткой страниц прокручивает и вкладки.

В данном случае мы обращаемся к свойству tab.text и передаем в заголовки одинаковый текст – слово «TAB» —  с номером вкладки. Во второй половине урока модифицируем этот участок кода для передачи разного текста в заголовки вкладок.

Метод attach() связывает TabLayout с ViewPager2.

Запустим приложение на устройстве или эмуляторе и убедимся, что на экране добавились вкладки. Номер вкладки в заголовке соответствует номеру страницы на экране, вкладки прокручиваются синхронно со страницами на экране.



**Уникальные имена для вкладок**

А что, если нам нужно присвоить вкладкам уникальные имена? Давайте это реализуем.

Изменим код класса MainActivity:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package info.fandroid.viewpager2app    import android.os.Bundle  import androidx.fragment.app.FragmentActivity  import androidx.viewpager2.widget.ViewPager2  import com.google.android.material.tabs.TabLayout  import com.google.android.material.tabs.TabLayoutMediator    class MainActivity : FragmentActivity() {        private lateinit var adapter: NumberAdapter      private lateinit var viewPager: ViewPager2      private lateinit var tabLayout: TabLayout      private val tabNames: Array<String> = arrayOf(          "Первый",          "Второй",          "Третий",          "Четвертый",          "Пятый",          "Шестой",          "Седьмой",          "Восьмой",          "Девятый",          "Десятый",      )        override fun onCreate(savedInstanceState: Bundle?) {          super.onCreate(savedInstanceState)          setContentView(R.layout.activity\_main)            adapter = NumberAdapter(this)          viewPager = findViewById(R.id.pager)          viewPager.adapter = adapter            tabLayout = findViewById(R.id.tab\_layout)          TabLayoutMediator(tabLayout, viewPager) { tab, position ->              tab.text = tabNames[position]          }.attach()      }    } |

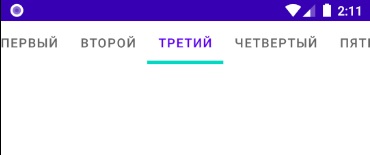
Мы создаем строковый массив tabNames с набором собственных имен для каждой вкладки. Затем присваиваем имена вкладкам: tab.text = tabNames[position]. Поскольку позиция вкладки служит нам индексом для обращения к элементам массива, имена в массиве должны располагаться в том же порядке.

Мы здесь создали массив имен только из десяти элементов, и во избежание ошибки нужно также ограничить десятью и количество вкладок. Мы определили количество вкладок на прошлом уроке в классе NumberAdapter, в методе getItemCount() — проверьте и исправьте:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package info.fandroid.viewpager2app    import android.os.Bundle  import androidx.fragment.app.Fragment  import androidx.fragment.app.FragmentActivity  import androidx.viewpager2.adapter.FragmentStateAdapter    class NumberAdapter(fragment: FragmentActivity) : FragmentStateAdapter(fragment) {      override fun getItemCount(): Int = 10        override fun createFragment(position: Int): Fragment {          val fragment = NumberFragment()          fragment.arguments = Bundle().apply {              putInt(ARG\_OBJECT, position + 1)          }          return fragment      }  } |

После запуска приложения на устройстве мы видим экран с десятью прокручиваемыми вкладками и уникальными именами заголовков.



**Отключение капса в заголовках вкладок**

Заголовки по умолчанию отображаются капсом. Если вам нужно, чтобы они отображались как обычный текст, нужно создать собственный стиль в файле themes.xml:



|  |  |
| --- | --- |
| 1  2  3  4 | <style name="MyCustomTextAppearance" parent="TextAppearance.Design.Tab">      <item name="textAllCaps">false</item>      <item name="android:textAllCaps">false</item>  </style> |

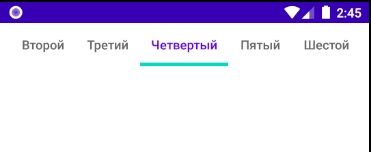
Это вернет текст в нормальное состояние. Почему нужно указывать сразу два свойства? Это связано с особенностями работы разных версий библиотек материального дизайна.

Теперь созданный стиль нужно применить компоненту TabLayout в activity\_main.xml:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <com.google.android.material.tabs.TabLayout      android:id="@+id/tab\_layout"      app:tabMode="scrollable"      app:tabIndicatorColor="@color/teal\_200"      app:tabIndicatorHeight="4dp"      android:layout\_width="match\_parent"      android:layout\_height="wrap\_content"      app:layout\_constraintEnd\_toEndOf="parent"      app:layout\_constraintStart\_toStartOf="parent"      app:layout\_constraintTop\_toTopOf="parent"      app:tabTextAppearance="@style/MyCustomTextAppearance"/> |

Вуаля:



Таким образом можно создавать собственные стили и кастомизировать не только текст, но и другие свойства вкладок.

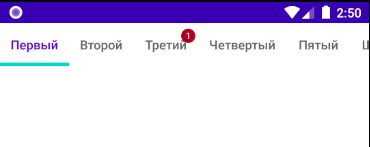
**Бейджи для вкладок**

Также вкладкам можно устанавливать бейджи, для этого измените код TabLayoutMediator  в классе MainActivity:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | TabLayoutMediator(tabLayout, viewPager) { tab, position ->      tab.text = tabNames[position]        if (position == 2) {          val badge = tab.getOrCreateBadge()          badge.number = 1      }    }.attach() |

Здесь мы в заголовок третьей вкладки добавляем бейдж с числом 1. Удалить бейджи можно методом tab.removeBadge().



**Иконки заголовков вкладок**

Можно также установить иконки в заголовках вкладок, вместо текста или вместе с ним.

Добавим в MainActivity массив со ссылками на иконки:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | private val tabNumbers: Array<Int> = arrayOf(          R.drawable.baseline\_looks\_one\_black\_48,          R.drawable.baseline\_looks\_two\_black\_48,          R.drawable.baseline\_looks\_3\_black\_48,          R.drawable.baseline\_looks\_4\_black\_48,          R.drawable.baseline\_looks\_5\_black\_48,          R.drawable.baseline\_looks\_6\_black\_48  ) |

Иконки. Вы можете взять их стандартные картинки или скачать самрстоятельно.

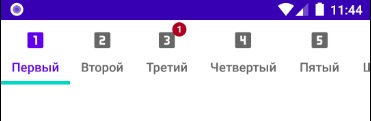
Поскольку иконок всего шесть, во избежание ошибки измените на шесть количество вкладок в методе getItemCount() класса NumberAdapter.

Теперь можно установить иконки для вкладок методом tab.setIcon(tabNumbers[position]):



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | TabLayoutMediator(tabLayout, viewPager) { tab, position ->      tab.text = tabNames[position]          tab.setIcon(tabNumbers[position])        if (position == 2) {          val badge = tab.getOrCreateBadge()          badge.number = 1      }    }.attach() |

Вот что должно получиться при запуске приложения:



Если удалить или закомментировать строку tab.text = tabNames[position], то в заголовках вкладок останутся только иконки:

