Приложение с вкладками – TabLayout с ViewPager2

В предыдущем задании мы познакомились с ViewPager2 и создали андроид-приложение, в котором можно листать экраны свайпом вправо или влево. В этом задании добавим в верхней части экрана вкладки, которые будут содержать заголовки и индикатор экрана, на котором находится пользователь в данный момент, а также рассмотрим некоторые их свойства и способы оформления.

1. Добавление TabLayout в макет разметки экрана
2. Реализация вкладок в MainActivity
3. Уникальные имена для вкладок
4. Отключение капса в заголовках вкладок
5. Бейджи для вкладок
6. Иконки заголовков вкладок

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:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.andro
id.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:

```
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

Чтобы присвоить вкладкам уникальные имена, проведем следующие изменения:

Изменим код класса MainActivity:

```
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:

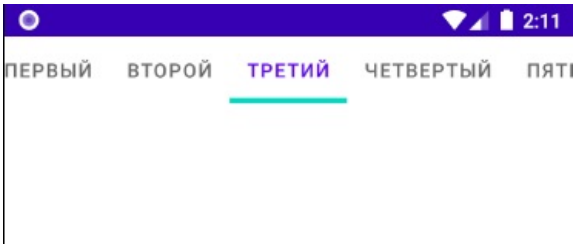
```
package info.fandroid.viewpager2app

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:

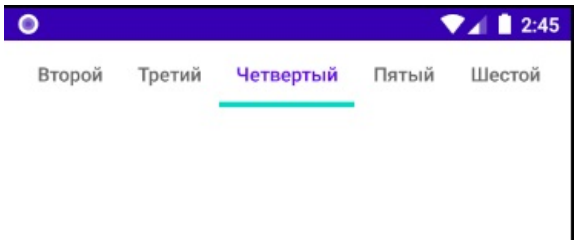
```
<style name="MyCustomTextAppearance" parent="TextAppearance.Design.Tab">
    <item name="textAllCaps">false</item>
    <item name="android:textAllCaps">false</item>
</style>
```

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:

```
<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"/>
```

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



Бейджи для вкладок

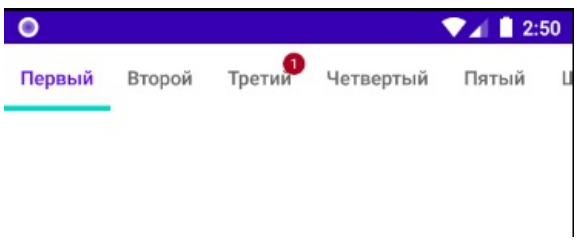
Также вкладкам можно устанавливать бейджи, для этого измените код TabLayoutMediator в классе MainActivity:

```
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:

```
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 были предварительно скачаны с сайта материального дизайна и сохранены в папку res/drawable нашего проекта.

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]):

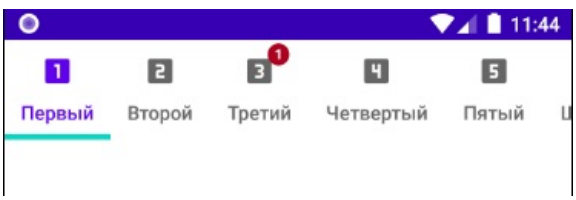
```
TabLayoutMediator(tabLayout, viewPager) { tab, position ->
    tab.text = tabNames[position]


    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: